NR INDEKSU 353965
PL ISSN 0860-1674

12

Z MIKROKOMPUTEREM NA TY

Bajtek

MIESIĘCZ[illegible] GRUDZIEŃ 1988 CENA 150 ZŁ



KONKURS KOLEJOWY!

MINI PAKIET GRAFICZNY DLA AMSTRADA

POŻYTECZNE PROCEDURY

LISTA ROZKAZÓW Z 80

LICZBY ZESPOLONE

HACKER

scan by Dimetylo
www.cconline.com.pl
Classic Computers Online

SPOSÓB NA WIRUSA

Wirus komputerowy jest w dalszym ciągu tematem nr 1 w rozmowach ludzi zajmujących się profesjonalnie komputerami. Ponieważ zalicza się do nich również zespół „Bajtka", więc z wątpliwą satysfakcją mam okazję zakomunikować, że również do naszego redakcyjnego PC przedostał się już wirus. Wprawdzie zbytnio nie narozrabiał, ale psuje nam krew i zabiera czas.

Zanim ten numer dotrze do Czytelników, zapewne damy już sobie z „naszym" wirusem radę.

Wyszło przy okazji na jaw, że zacofanie cywilizacyjne Polski, przejawiające się brakiem sieci komputerowych, okazało się w tym jednym przypadku plusem. Bo aż włosy nam się jeżą na głowie na myśl, że nasze komputery byłyby sprzęgnięte w sieć! Futurolodzy wymyślili kiedyś pojęcie „szansy spóźnionego przybysza". Otóż w tym jednym, jedynym przypadku, rzeczywiście coś z tego zapóźnienia mamy. Choć, szczerze mówiąc satysfakcja z tego wątpliwa.

Japończycy natomiast wpadli w panikę. W ich naszpikowanym komputerami i oplecionym pajęczyną teleinformatycznych połączeń kraju, pojawienie się wirusa zaczęło pachnieć narodową katastrofą. Żeby jej zapobiec, japońskie ministerstwo Przemysłu i Handlu Zagranicznego słynne MITI, powołało specjalną grupę fachowców, dysponującą nadzwyczajnymi uprawnieniami.

Jak informuje pan Sejdzi Hagiwara, główny spec od komputerów w MITI, grupa ta dostała za zadanie zidentyfikować wirusa do końca 1988 roku, wyprodukować „lekarstwo" — czyli program-zabójcę oraz dokonać skutecznej „dezynseksji" sieci komputerowej Kraju Kwitnącej Wiśni.

Równocześnie, ponieważ Japończycy nie są takimi naiwniakami, żeby wierzyć w jakikolwiek monopol, nawet naukowy, powołano konkurencyjną grupę i postawiono jej to samo zadanie. Żeby było ciekawiej, grupę tę wyłoniono spośród komputerowej czołówki korporacji „Nixon Denki", w której sieciach po raz pierwszy w Japonii wykryto wirusa. Było to w sierpniu 1988 roku. Od tego czasu informacje o wirusie napłynęły również wielu innych firm. Ale pan Sejdżi Hagiwara jest przekonany, że systemowa, prowadzona z rozmachem akcja przeciwwirusowa powinna szybko dać efekty.

Za to w Stanach Zjednoczonych wirus komputerowy już tak się rozpowszechnił, że Ministerstwo Obrony USA musiało podjąć decyzję o zablokowaniu wszystkich linii teleinformatycznych łączących jawną sieć komputerową Pentagonu z siecią naukowo-badawczą USA. „Podjęto tę trudną decyzję — stwierdził „New Jork Times" — gdy stało się jasno, że właśnie tą drogą do komputerów w wielu strategicznych obiektach obronnych USA, przedostały się nieznane programy".

Wścibscy dziennikarze amerykańscy ustalili, że konkretnie zablokowano sieć „Milnet", sprzęgającą setki komputerów Pentagonu, korporacji i uniwersytetów całego kraju. „Milnet" kilkoma kanałami wchodzi do potężnej, już czysto wojskowej, sieci „Arpnet", tej właśnie, która stała się główną ofiarą ataku wirusa na początku listopada 1988. Zdezorganizowana została wówczas praca około 6000 komputerów. Poniesione przez Pentagon straty ocenia się na 95 milionów dolarów.

Fachowcy z Pentagonu przystąpili obecnie do szczegółowego sprawdzania wszystkich elementów zabezpieczenia swoich sieci komputerowych, aby maksymalnie wyeliminować możliwość przeniknięcia do nich jakichkolwiek intruzów. Jeśli natomiast chodzi o poszukiwania nieznanego programisty — który sparaliżował sieć, to, jak donosi „New Jork Times", bardzo sprytnie zatarł on wszystkie ślady.

Waldemar Siwiński

ZA MIESIĄC:

- Dowiecie się, ile bitow ma „Bajtek"
- Zwiedzicie wraz z nami Kolońskie targi „Orgatechnik" oraz wystawę „Informacja 89"
- Poznacie dalsze tajniki monitorów
- Przekonacie się o tym, jak pożytecznym urządzeniem jest modem
- Ponadto w numerze 1/89 znajdziecie m.in.: mapę „Universal Hero", kalendarz-niespodziankę, mikrociekawostki i wiele innych atrakcji

Wszystkim naszym Czytelnikom wspaniałego, optymistycznego roku 1989, najciekawszych programów i najlepszych komputerów życzy „Bajtek"

Bajtek

„BAJTEK" — MIESIĘCZNY DODATEK DO „SZTANDARU MŁODYCH"

ADRES: 00-687 Warszawa, ul. Wspolna 61. Tel. 21-12-05 Przewodniczący Rady Redakcyjnej: Jerzy Domański — redaktor naczelny „Sztandaru Młodych".

ZESPÓŁ REDAKCYJNY: Waldemar Siwiński (z-ca redaktora naczelnego „SM" — kierownik zespołu „Bajtka"), Grzegorz Onichimowski (sekretarz redakcji „Bajtka"), Roman Poznański (kierownik działu klanów), Krzysztof Czernek, Sławomir Gajda (red. techniczny), Andrzej Pilaszek, Sławomir Polak, Wanda Roszkowska (opr. graficzne), Kazimierz Treger, Marcin Waligórski, Roman Wojciechowski. Zdjęcia w numerze: Leopold Dzikowski

klany redagują:
Commodore — Klaudiusz Dybowski, Dominik Falkowski
Amstrad-Schneider — Jonasz Mayer
Spectrum — Marcin Przasnyski
Atari — Wojciech Zientara, Sergiusz Piotrowski

Fotoskład — Tadeusz Olczak,
Montaż offsetowy — Grażyna Ostaszewska,
Korekta — Maria Krajewska, Zofia Wółtańska

WYDAWCA: RSW „Prasa-Książka-Ruch" Młodzieżowa Agencja Wydawnicza, al. Stanów Zjednoczonych 53, 04-028 Warszawa. Telefony: Centrala 13-20-40 do 49, Redakcja Reklamy 13-20-40 do 49 w. 403, 414.
Cena 150 zł.
Skład techniką CRT-200, przygotowalnia offsetowa i druk: PRASOWE ZAKŁADY GRAFICZNE RSW „PRASA-KSIĄŻKA-RUCH" w Ciechanowie, ul. Sienkiewicza 51.
Nr zlecenia 055528 n. 150.000 egz.
U-113









14 października był nie tylko „Dniem Bajtka" podczas tegorocznej wystawy MikroEXPO 88. Był to również — a właściwie przede wszystkim — „Dzień Nauczyciela", święto wszystkich pedagogów. Im więc właśnie postanowiliśmy poświęcić najwięcej uwagi, tym bardziej, że myślą przewodnią całej wystawy były edukacyjne zastosowania komputerów.

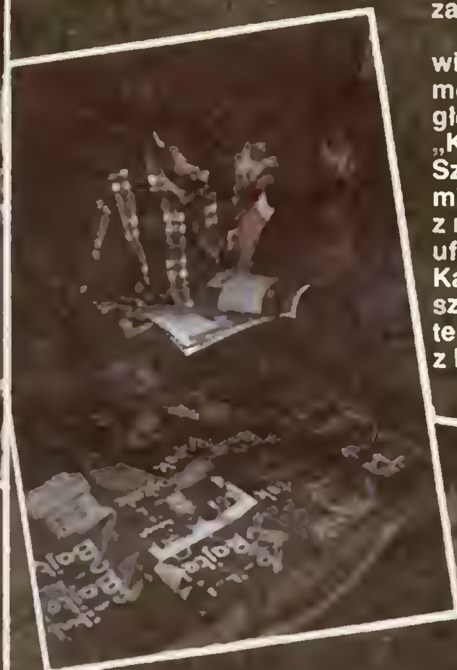
Nauka nauką, ale w takim dniu należy się trochę odpoczynku, kwiatek dla Pani i Pana, a także — jakby inaczej — piękna laurka.

Komu zabrakło czasu lub zdolności, by namalować w domu laurkę przy pomocy pędzla i farb, mógł to zrobić na miejscu, korzystając z komputera i drukarki. Efekty były wspaniałe, o czym zapewniali nas obdarowani.

Ażeby nasi szanowni pedagodzy czuli się jak u siebie w szkole, zorganizowaliśmy — korzystając z pomocy PZ Karen — prawdziwą klasę szkolną z ławkami, tablicą i... komputerami na każdej ławce. Trzeba przyznać, że zainteresowanie nauczycieli komputerową klasą co najmniej dorównywało zainteresowaniu uczniów.

Miłym i oczekiwanym przez wielu naszych Czytelników momentem było losowanie głównej nagrody w „Konkursie z Szesnaściorem", mikrokomputera Atari 65 XE z magnetofonem, ufundowanego przez PZ Karen. Przypomnijmy, że szczęśliwym posiadaczem tego sprzętu został Artur Zeh z Lublina.

(rp)





DLA NASZEJ PANI...

MUZYKA W PRZERWANIACH

Dzięki wiadomościom zawartym w „Bajtku" przestałem bać się przerwań i napisałem program w języku maszynowym, który jest wykonywany co 1/50 sekundy w czasie przerwania synchronizacji pionowej.

Opracowana przeze mnie procedura może wykonywać proste melodie (do 93 nut), lecz nie przeszkadza w normalnej pracy Basica. Mozna więc wykorzystać ją do uzyskania podkładu muzycznego we własnym programie, zaś przy pisaniu programu może ona umilić pracę.

Dla uzyskania takiego efektu trzeba wpisać zamieszczony program i uruchomić go. Program ten jest potem zbędny i można go skasować instrukcją NEW.

Przerwanie pracy programu następuje po naciśnięciu klawisza RESET lub po wpisaniu POKE 548, 138: POKE 549, 194.

Ponowne włączenie muzyki uzyskuje się po wpisaniu POKE 548, 0: POKE 549, 6.

Jeśli chcemy uzyskać inną melodię, to trzeba wpisać do komórki 1541 liczbę nut pomnożoną przez 2 (maksymalnie 186), a do komórki 1582 wielkość pauzy między nutami (maksymalnie 255). Teraz jest to odpowiednio liczba 132 w wierszu 90 i trzecia liczba 0 w wierszu 100. Następnie trzeba zmienić zawartość instrukcji DATA od wiersza 120. Kolejno trzeba tam wpisać dla każdej nuty wysokość dźwięku (wartości takie jak w instrukcji SOUND) i czas jego trwania. Na końcu w wierszu 70 wpisujemy graniczną wartość pętli FOR/NEXT odpowiednią dla liczby nut wprowadzonych do instrukcji DATA.

Dla Czytelników zainteresowanych programowaniem w języku maszynowym podaję dodatkowo źródłowy wydruk procedury w asemblerze MAC/65.

Maciej Rempiński (lat 15)

```
CT  10 REM MUZYKA W PRZERWANIACH
AD  20 REM Maciej Rempinski
AK  30 REM Copyright (c) Bajtek
BB  40 REM
WJ  50 FOR I=1536 TO 1604:READ A:POKE I,A:
    NEXT I
DP  60 POKE 203,0:POKE 204,0:POKE 205,0
ZQ  70 FOR I=1605 TO 1736:READ A:POKE I,A:
    NEXT I
FC  80 POKE 548,0:POKE 549,6
QH  90 DATA 8,72,166,203,224,132,240,23,16
    9,69,6,141,0,210,169,234,141,1,210,165
    ,205,221,70,6,240,12,230,205
MX  100 DATA 76,64,6,169,0,133,203,76,84,6
    ,169,0,141,1,210,165,204,201,0,240,5,2
    30,204,76,64,6,169,0,133,204
LE  110 DATA 133,205,230,203,230,203,40,10
    4,76,136,194
LS  119 REM DANE DLA MELODII
PD  120 DATA 40,7,45,7,47,7,0,7,40,7,0,7,2
    9,7,0,7,23,7,0,7,26,7,0,7,26,7,0,20,29
    ,7,31,7,35,7,0,7,29,7,0,7,31,7
DY  130 DATA 0,7,35,7,0,7,40,7,0,34,35,7,0
    ,7,45,7,0,34,40,7,0,7,47,7,0,34,45,7,4
    8,7,53,7,0,7,53,7,0,7,35,7,0,7
MP  140 DATA 35,7,0,7,40,7,0,34,35,7,0,7,4
    5,7,0,34,40,7,0,7,47,7,0,34,45,7,48,7,
    53,7,0,7,53,7,0,7,47,7,0,7
XH  150 DATA 47,7,0,7,60,7,0,27
```

```
0100 ;Muzyka w przerwaniach
0110 ;Maciej Rempinski
0120 ;Copyright (c) Bajtek
0130 ;
0140 AUDF0 = $D200
0150 AUDC0 = $D201
0160 EXITVBL = $C28A
0170 LICZNIK = 203
0180 PAUZA = 204
0190 DZWIEK = 205
0200 NUTY =  $0645
0210 ;
0220      *=  $0600
0230 ;
0240      PHP
0250      PHA
0260      LDX LICZNIK
0270      CPX #132
0280      BEQ SKOK1
0290      LDA NUTY,X
0300      STA AUDF0
0310      LDA #234
0320      STA AUDC0
0330      LDA DZWIEK
0340      CMP NUTY+1,X
0350      BEQ SKOK2
0360      INC DZWIEK
0370      JMP END
0380 SKOK1 LDA #0
0390      STA LICZNIK
0400      JMP END
0410 SKOK2 LDA #0
0420      STA AUDC0
0430      LDA PAUZA
0440      CMP #0
0450      BEQ SKOK3
0460      INC PAUZA
0470      JMP END
0480 SKOK3 LDA #0
0490      STA PAUZA
0500      STA DZWIEK
0510      INC LICZNIK
0520      INC LICZNIK
0530 END  PLA
0540      PLP
0550      JMP EXITVBL
```

ZAMALOWYWANIE

Istniejąca w Action! procedura Fill () wypełnia pole w prawo od rysowanej linii (jak XIO 18 w Basicu). Utrudnia to znacznie zamalowywanie skomplikowanych kształtów. Wady tej nie posiada procedura Paint ().

Procedura Paint () jest oparta na algorytmie opisanym w „Młodym Techniku" 11/85. Potrzebuje ona do pracy znacznego obszaru pamięci, co ogranicza jej wykorzystanie do niezbyt dużych obszarów. Mimo tego utrudnienia jest ona bardzo pożyteczna i na pewno znajdzie miejsce w bibliotece każdego programisty. Dołączona do niej procedura Demo () umożliwia sprawdzenie działania Paint ().

Andrzej Postrzednik

```
;Zamalowywanie obszaru
;Andrzej Postrzednik
;Copyright (c) Bajtek

MODULE

  CARD ARRAY xw(1000)
  BYTE ARRAY yw(1000)


CARD FUNC Oko(CARD i,x BYTE y)

  IF Locate(x,y)=0 THEN
    Plot(x,y) i==+1
    xw(i)=x yw(i)=y
  FI
RETURN(i)


PROC Paint(CARD x BYTE y)

 CARD i=[0]

  WHILE 1
   DO
    i=Oko(i,x,y)
    IF i=0 THEN RETURN FI
    x=xw(i) y=yw(i) i==-1
    x==-1 i=Oko(i,x,y)
    x==+2 i=Oko(i,x,y)
    x==-1 y==-1 i=Oko(i,x,y)
    y==+2
   OD
RETURN


PROC Demo()

  Graphics(7) color=1
  Plot(10,10) DrawTo(80,20)
  DrawTo(60,60) DrawTo(30,50)
  DrawTo(20,30) DrawTo(10,10)
  color=2
  Paint(45,20)
  PrintE("Nacisnij START")
  WHILE Peek(53279)<>6 DO OD
RETURN
```

POŻYTECZNE PROCEDURY

Pisanie programu w języku Action! polega przede wszystkim na umiejętnym zestawieniu odpowiednich procedur. Każdy szanujący się programista musi więc mieć bogatą bibliotekę tych procedur. Aby nie wymyślić wszystkiego od nowa, otwieramy kącik, w którym będziemy publikować proste procedury. Liczymy także na wsparcie przez Czytelników.

Rozpoczniemy od dwóch prostych funkcji matematycznych — ABS i SGN. Pierwsza z nich zwraca wartość bezwzględną podanej liczby. Wystarczy więc tylko sprawdzić znak argumentu i zmienić go na przeciwny, jeśli jest ujemny.

```
INT FUNC Abs(INT k)

   IF k<0 THEN RETURN(-k) FI
RETURN(k)
```

Funkcja SGN zwraca wartość określającą znak podanego argumentu. Jeżeli argument jest ujemny, to wynikiem będzie liczba −1, a gdy dodatni, to +1. Dla argumentu zerowego rezultatem będzie oczywiście zero.

```
INT FUNC Sgn(INT k)

   IF k<0 THEN RETURN(-1)
   ELSEIF k>0 THEN RETURN(1)
   FI
RETURN(0)
```

Korzystając z pokazanej wyżej zasady możemy zbudować funkcję odczytującą poziomy ruch joysticka. Przy ruchu w lewo funkcja zwróci wartość −1, w prawo +1, a w położeniu neutralnym 0. Można tu zastosować dwa warianty rozwiązania. W pierwszym aktualne położenie joysticka odczytujemy korzystając z bibliotecznej funkcji Stick(n), gdzie n jest numerem joysticka (0 lub 1). Ponieważ jest to dokładnie to samo co Peek($278+n), można zastosować bezpośredni odczyt wartości z rejestru RAM. Użyjemy w tym celu dwuelementowej tablicy. Samo ustalenie kierunku ruchu jest bardzo proste. w rejestrze joysticka przy ruchu w lewo mogą się znaleźć wartości 9, 10 lub 11, przy ruchu w prawo 5, 6 lub 7, zaś w położeniu środkowym (w poziomie) — 13, 14 lub 15. Ostatecznie procedura ma następującą postać

```
INT FUNC HStick(BYTE joy)

BYTE ARRAY port(2)=$278

   IF port(joy)<8 THEN RETURN(1)
   ELSEIF port(joy)>12 THEN RETURN(0)
   FI
RETURN(-1)
```

Nieco bardziej skomplikowana będzie funkcja odczytująca pionowe położenie joysticka. W odpowiednim rejestrze odczytamy dla ruchu w górę wartości 6, 10 lub 14, a dla ruchu w dół 5, 9 lub 13, a w położeniu neutralnym 7, 11 lub 15. zauważmy jednak, że w każdym z tych przypadków ważne są tylko dwa najmłodsze bity rejestru. Jeżeli uwzględnimy tylko te bity, to uzyskamy odpowiednio wartości 1, 2, 1 i 3. Teraz trzeba to wpisać do procedury i gotowe.

```
INT FUNC VStick(BYTE joy)

BYTE ARRAY port(2)=$278
BYTE i

   i=port(joy)&3
   IF i=1 THEN RETURN(1)
   ELSEIF i=3 THEN RETURN(0)
   FI
RETURN(-1)
```

Kolejne procedury będą dotyczyły różnych operacji graficznych. Zaczniemy od narysowania prostokąta. Jako parametry wejściowe podamy współrzędne dwóch przeciwległych wierzchołków. Ich położenie względem siebie jest w tym przypadku nieistotne. Ponieważ maksymalna wartość współrzędnej pionowej wynosi 191 (tryb ósmy bez okna tekstowego), to wystarczy zmienna typu BYTE. Dla współrzędnej poziomej konieczna jest jednak zmienna typu CARD, gdyż w trybie ósmym może ona osiągnąć wartość 319. Do wykreślenia samego prostokąta potrzeba tylko cztery linie, więc procedura jest bardzo prosta. Dodatkowo zastosujemy w niej piąty parametr, który określi kolor rysowanej ramki.

```
PROC Frame(CARD x1 BYTE y1
           CARD x2 BYTE y2,c)

   color=c Plot(x1,y1)
   DrawTo(x2,y1) DrawTo(x2,y2)
   DrawTo(x1,y2) DrawTo(x1,y1)
RETURN
```

Mając już prostokąt będziemy chcieli wypełnić go kolorem. W takim przypadku nie wystarczy prosta zamiana ostatniej procedury DrawTo na Fill. Procedura wypełniania zawarta w systemie operacyjnym Atari zawsze wypełnia linię obrazu w prawo od narysowanego punktu. Musimy więc sprawdzić, czy linia ograniczająca wypełniane pole znajduje się z prawej strony, a jeśli nie, to zamienić wartości współrzędnych poziomych. Druga instrukcja warunkowa IF uniemożliwia błędne zamalowanie obrazu, gdy obie pionowe linie się pokrywają. Ponadto kolor do wypełniania nie musi być taki sam jak kolor do rysowania. Wprowadzimy więc dodatkowy parametr, który będzie go definiował.

```
PROC Box(CARD x1 BYTE y1
         CARD x2 BYTE y2,c,f)

CARD aux
BYTE fcolor=$2FD

   color=c fcolor=f
   IF x1>x2 THEN
      aux=x1 x1=x2 x2=aux FI
   Plot(x1,y1) DrawTo(x2,y1)
   DrawTo(x2,y2) DrawTo(x1,y2)
   IF x1<>x2 THEN Fill(x1,y1) FI
RETURN
```

Teraz przyszedł czas na narysowanie okręgu. To zadanie jest znacznie bardziej skomplikowane, nie będziemy więc rozważać go szczegółowo. Wystarczy powiedzieć, że zastosowany został algorytm rysowania okręgu w ośmiokierunkowej symetrii zmodyfikowany przez usunięcie funkcji pierwiastkowania, której nie ma w Action!. Wyprowadzenie tego algorytmu było opisane w „Komputerze" 9/86. Gotowa procedura jest pokazana poniżej. Jej parametrami wejściowymi są współrzędne środka okręgu, jego promień i kolor, którym będzie wykreślony.

```
PROC Circle(CARD sx BYTE sy,r,c)

INT dx,dy,dxy,pdx,pdy

   color=c
   dx=0 pdx=r pdy=0
   DO
      dy=dx+pdy+pdy+1
      dxy=dy-pdx-pdx+1
      Plot(sx+pdx,sy+pdy)
      Plot(sx-pdx,sy+pdy)
      Plot(sx+pdx,sy-pdy)
      Plot(sx-pdx,sy-pdy)
      Plot(sx+pdy,sy+pdx)
      Plot(sx-pdy,sy+pdx)
      Plot(sx+pdy,sy-pdx)
      Plot(sx-pdy,sy-pdx)
      dx=dy pdy==+1
      IF Abs(dxy)+0<Abs(dy) THEN
         dx=dxy pdx==-1
      FI
   UNTIL pdy>pdx
   OD
RETURN
```

W ten sposób zaczynając od matematyki doszliśmy do rysunku. Korzystając z powyższych procedur można napisać prosty program graficzny obsługiwany przy pomocy joysticka. Ale to już samodzielnie — teraz Ty, Czytelniku.

Wojciech Zientara

NAMER

Jedną z większych wad komputera Atari jest brak możliwości nazwania programów i gier.

Napisany przeze mnie program eliminuje tę wadę. Przystosowany jest do nazywania programów napisanych w asemblerze i ładowanych przez START i OPTION. Jest całkowicie napisany w Basicu (Ładowanie-CLOAD, uruchamianie — RUN). Zaraz po uruchomieniu odzywa się pojedyńczy „beep" sygnalizujący gotowość do wczytania nagłówka programu. Zaraz po nagłówku należy skopiować program lub grę (razem z loaderem jeśli taki jest).

Osobiście radzę skopiować program zaraz po pierwszym rekordzie nagłówka — nie czekając na rekord kończący. Po wczytaniu nazwy, program pyta się czy ma załadować daną np. grę. Jeśli tak, należy wcisnąć „T" a następnie START + OPTION i po usłyszeniu „beep" dowolny klawisz — program będzie ładowany. W przypadku decyzji odmownej komputer wraca na początek programu.

Program jest prosty i wydaje mi się, że nie wymaga szczegółowego omówienia.

```
NL  1 REM * NAMER * - Robert Pindera
PA  2 DIM B$(120)
PK  3 ? CHR$(125):OPEN #1,4,128,"C:":INPUT
     #1;B$:? B$:POSITION 1,1:? "PROGRAM:":
     CLOSE #1
TF  4 POSITION 3,15:? "Czy mam wczytac ten
     program? [T/N]":OPEN #3,4,0,"K:":GET
     #3,A:CLOSE #3:IF A<>84 THEN 3
TR  5 POSITION 5,17:? "Wcisnij START i OPT
     ION..."
YC  6 IF PEEK(53279)<>2 THEN 6
UR  7 A=USR(58487)
```

Najwygodniej będzie nagrać go na początku każdej kasety i później szukać ewentualnego programu do wczytania.

Podaję teraz sposób tworzenia nagłówka — należy napisać
OPEN#1,8,128,„C:":
PRINT#1, „nazwa": CLOSE#1
po czym wcisnąć RETURN. Usłyszymy podwójny „beep" i po wciśnięciu dowolnego klawisza komputer wgra nagłówek.

Robert Pindera

KLAN ATARI

LICZBY ZESPOLONE

Liczby zespolone są mało znanym uczniom szkół średnich działem matematyki (praktycznie tylko uczniom klas o profilu matematyczno-fizycznym). Tym jednak, którzy poznają liczby zespolone przydatny może być mój program.

Realizuje on najczęściej wykonywane działania na liczbach zespolonych takie jak: mnożenie, dzielenie, dodawanie, odejmowanie, potęgowanie zespolone, pierwiastkowanie, ponadto wylicza logarytm naturalny, funkcje sinus i cosinus, exponent, moduł i liczbę sprzężoną oraz zamienia postać algebraiczną na trygonometryczną i odwrotnie. W opcjach „DODAWANIE" i „MNOŻENIE" program pyta o ilość liczb, na których będzie wykonywanie działania. Umożliwia to dodawanie lub mnożenie wielu liczb zespolonych (jednak nie więcej niż 17).

Program mój został częściowo oparty na algorytmach zawartych w książce „2^5 numerycznych programów w języku BASIC" Andrzeja Witowskiego i Jakuba Tatarkiewicza.

Jeszcze kilka uwag na koniec. Ze względu na możliwości obliczeniowe Atari nie wszystkie dane będą akceptowane przez program, o czym użytkownik jest informowany odpowiednim komunikatem. Większość wyników podawana jest z dokładnością do 0.001. W opcjach zmiany postaci z algebraicznej na trygonometryczną i odwrotnie ułamki za kropką oznaczają dziesiętne i setne stopnia, a nie minuty i sekundy.

Dariusz Tabaczyński

```
KI  10 REM LICZBY ZESPOLONE
OH  20 REM Dariusz Tabaczynski
ET  30 REM Copyright 1988 Bajtek
OM  100 REM
AR  110 DIM A(10),B(10),C$(1),D$(1):SETCOLOR 2,0,1:SETCOLOR 1,0,11:POKE 82,2
AK  120 REM MENU
XA  130 ? "↓[1] - DODAWANIE","+[***** LICZBY ZESPOLONE *****]","↓[2] - ODEJMOWANIE [*****]";
OH  140 ? "↓[3] - MNOZENIE" MENU ":? ,," ":? "[4] - DZIELENIE"
KG  145 ? ,,"+"
XN  150 ? "[5] - MODUL I LICZBA SPRZEZONA":? "↓[6] - LOGARYTM NATURALNY LICZBY ZESPOL."
EH  160 ? "↓[7] - EXPONENT":? "↓[8] - SINUS I COSINUS LICZBY ZESPOLONEJ":? "↓[9] - POTEGA ZESPOLONA":? "↓[A] - ZAMIANA NA ";
MR  170 ? "POSTAC TRYGONOMETRYCZNA":? "[B] - ZAMIANA NA POSTAC ALGEBRAICZNA":? "↓[C] - PIERWIASTEK Z LICZBY ZESPOLONEJ"
LR  180 CLOSE #1:OPEN #1,4,0,"K:":TRAP 180:POSITION 11,23:? "[WYBIERAJ]";:GET #1,C:CLOSE #1
BT  190 ON C-47 GOTO 220,280,310,360,400,440,470,500,570,610
FW  200 ON C-59 GOTO 760,180,880
.T  210 GOTO 180
FF  220 REM DODAWANIE
BY  230 ? "K":? "[**** DODAWANIE ****]":? :TRAP 220:? "Ile liczb zespolonych chcesz dodac";:INPUT N:?
MX  240 GOSUB 1270:GOSUB 1050:GOSUB 260:GOSUB 270:IF N=2 THEN K=8:GOSUB 1290:? "↑↑  Z1+Z2":GOTO 1100
MM  250 K=10+(N>=10):GOSUB 1290:? "↑↑Z1+...+Z";N:GOTO 1100
NO  260 A=0:FOR L=1 TO N:A=A+A(L):NEXT L:A(N+1)=A:RETURN
AN  270 B=0:FOR L=1 TO N:B=B+B(L):NEXT L:B(N+1)=B:RETURN
KN  280 REM ODEJMOWANIE
TO  290 ? "[**** ODEJMOWANIE ****]":? :N=2:GOSUB 1050:A(3)=A(1)-A(2):B(3)=B(1)-B(2):K=6:GOSUB 1290
OV  300 ? "↑↑Z1-Z2":GOTO 1100
HT  310 REM MNOZENIE
ZO  320 ? "K":? "[**** MNOZENIE ****]":? :TRAP 310:? "Ile liczb zespolonych chcesz mnozyc";:INPUT N:?
WF  330 GOSUB 1270:GOSUB 1050:A(N+1)=A(1)*A(2)-B(1)*B(2):B(N+1)=A(1)*B(2)+A(2)*B(1):IF N>2 THEN GOSUB 1220
FN  340 IF N=2 THEN K=9:GOSUB 1290:? "↑↑ Z1*Z2":GOTO 1100
EZ  350 K=10+(N>=10):GOSUB 1290:? "↑↑Z1*...*Z";N:GOTO 1100
KK  360 REM DZIELENIE
PC  370 ? "K":N=2:? "[**** DZIELENIE ****]":? :GOSUB 1050:IF A(2)=0 AND B(2)=0 THEN 390
OL  380 C=A(2)*A(2)+B(2)*B(2):A(3)=(A(1)*A(2)+B(1)*B(2))/C:B(3)=(A(2)*B(1)-A(1)*B(2))/C:K=9:GOSUB 1290
LZ  385 ? "↑↑   Z1/Z2":GOTO 1100
LI  390 ? "↓DZIELENIE PRZEZ ZERO JEST NIEWYKONALNE":FOR L=1 TO 800:NEXT L:GOTO 120
SO  400 REM MODUL I L.SPRZEZONA
HB  410 ? "[**** MODUL I SPRZEZONA ****]":? :N=1:GOSUB 1050:A(2)=A(1):B(2)=-B(1):K=9:GOSUB 1290:? "↑↑↑▸  _"
BS  420 GOSUB 1140:? "▸↓↓|Z|=";:IF MOD2=0 THEN ? MOD:GOTO 1100
KK  430 ? "SQR(";MOD2;")":GOTO 1100
DZ  440 REM LOGARYTM NATURALNY
SX  450 ? "[**** LOGARYTM NATURALNY ****]↓":N=1:GOSUB 1050:IF A(1)=0 AND B(1)=0 THEN 1250
QB  460 N=2:GOSUB 1170:K=9:GOSUB 1290:? "↑↑   LN(Z)":GOTO 1100
QL  470 REM EXPONENT
GR  480 ? "[**** EXPONENT ****]↓":N=1:GOSUB 1050:TRAP 1260:N=2:GOSUB 1200:K=9:GOSUB 1290
IJ  490 ? "↑↑  EXP(Z)":GOTO 1100
GT  500 REM SINUS I COSINUS
WF  510 ? "[**** SINUS I COSINUS ****]↓":N=1:GOSUB 1050:TRAP 1260:N=EXP(B(1)):N=(N+1/N)/2:A(2)=N*SIN(A(1))
DU  520 A(3)=N*COS(A(1))
YS  530 N=EXP(B(1)):N=(N-1/N)/2:B(2)=N*COS(A(1)):B(3)=-N*SIN(A(1)):K=9:GOSUB 1290:? "↑↑  SIN(Z)":? "↓  COS(Z)=";:L=3
VE  540 IF ABS(A(3))>=1.0E-03 THEN ? SGN(A(3))*(INT(1000*SGN(A(3))*A(3)+0.1))/1000;:X=3:GOTO 560
GW  550 IF A(3)<>0 THEN ? A(3);:X=3
PL  560 GOSUB 1340:GOTO 1100
VF  570 REM POTEGA ZESPOLONA
XK  580 ? "[**** POTEGA ZESPOLONA ****]↓":N=2:GOSUB 1050:TRAP 1260:IF A(1)=0 AND B(1)=0 THEN 1250
PV  590 N=3:GOSUB 1180:A(4)=A(2)*A(3)-B(2)*B(3):B(4)=B(2)*A(3)+A(2)*B(3):N=4:GOSUB 1200:N=2:K=4:GOSUB 1290
TX  600 POSITION 0,6:? "Z1^Z2":GOTO 1100
KY  610 REM POSTAC TRYGONOM.
MA  620 ? "[**** POSTAC TRYGONOMETRYCZNA ****]↓":N=1:GOSUB 1050:DEG :IF A(1)=0 AND B(1)=0 THEN 1250
MM  630 GOSUB 1140:A(2)=A(1)/MOD:B(2)=B(1)/MOD
CT  635 IF A(2)<>1 AND A(2)<>-1 THEN A(3)=-ATN(A(2)/SQR(1-A(2)*A(2)))+90:GOTO 660
MS  640 IF A(2)=1 THEN A(3)=0:GOTO 660
MN  650 A(3)=180
BY  660 IF B(2)<>1 AND B(2)<>-1 THEN B(3)=ATN(B(2)/SQR(1-B(2)*B(2))):GOTO 690
SC  670 IF B(2)=1 THEN B(3)=90:GOTO 690
UX  680 B(3)=270
YE  690 D$="":IF A(2)<0 THEN A(3)=180-A(3):D$="-"
HX  700 GOSUB 1300:A(3)=(INT(100*A(3)+0.1))/100:POSITION 0,5:? "Z=";
XI  710 IF MOD2>0 THEN ? "SQR(";(INT(100*MOD2+0.1))/100;")";:GOTO 730
MU  720 ? (INT(100*MOD+0.1))/100;
BM  730 ? "*(";D$;"COS(";A(3);")";
SO  735 IF B(3)>=0 THEN ? "+i*SIN(";(INT(100*B(3)+0.1))/100;"))           ":C$="":GOTO 750
MN  740 ? "-i*SIN(";(INT(100*ABS(B(3))+0.1))/100;"))           ":C$="-"
HN  750 ? ,"↓";C$;"SIN(";(INT(100*ABS(B(3))+0.1))/100;")=";B(2):? ,"↓";D$;"COS(";A(3);")=";A(2):GOTO 1100
IN  760 REM POSTAC ALGEBR.
GK  770 ? "[**** POSTAC ALGEBRAICZNA ****]↓ "," POSTAC:↓":? "▸ Z=R*(COS(Θ)+i*SIN(Θ))"
UW  780 TRAP 780:POSITION 4,7:? "[ ] MODUL R=";:INPUT MOD
KD  790 TRAP 790:POSITION 4,9:? "[ ]KAT (W STOPNIACH  0<=Θ<180 ) Θ=";:INPUT C:IF C<0 OR C>=180 THEN GOTO 790
JA  800 TRAP 800:POSITION 4,11:? "[ ] ZNAK FUNKCJI COSINUS (+/-) ";:INPUT C$:IF C$<>"-" AND C$<>"+" THEN 800
WP  810 TRAP 810:POSITION 4,13:? "[ ] ZNAK FUNKCJI SINUS (+/-) ";:INPUT D$:IF D$<>"+" AND D$<>"-" THEN 810
TC  820 TRAP 40000:IF C$="+" THEN C$=""
KM  830 DEG :A(1)=COS(C)*MOD:B(1)=SIN(C)*MOD:? "↓↓[* * * ZAMIANA * * *]↓↓":? "▸Z=";
KK  840 IF ABS(A(1))>=1.0E-03 THEN ? C$;SGN(A(1))*(INT(1000*SGN(A(1))*A(1)+0.1))/1000;:X=3:GOTO 860
EP  850 IF A(1)<>0 THEN ? C$;A(1);:X=3
FN  860 IF D$="-" THEN B(1)=-B(1)
GF  870 L=1:GOSUB 1350:? :GOTO 1100
WE  880 REM PIERWIASTEK
XO  890 ? "[**** PIERWIASTEK ****]↓":N=1:GOSUB 1050:DEG
JB  900 TRAP 900:POSITION 4,7:? "[ ]STOPIEN PIERWIASTKA (N=1,2,...,14)":? ,"▸N=";:INPUT C
FH  905 IF C<0 OR C>14 OR INT(C)<>C THEN 900
XJ  910 GOSUB 1140:R=MOD^(1/C):A(2)=A(1)/MOD:B(2)=B(1)/MOD
MY  920 IF A(2)<>1 AND A(2)<>-1 THEN A(3)=-ATN(A(2)/SQR(1-A(2)*A(2)))+90:GOTO 950
PE  930 IF A(2)=1 THEN A(3)=0:GOTO 950
WO  940 A(3)=180
NK  950 IF B(2)<>1 AND B(2)<>-1 THEN B(3)=ATN(B(2)/SQR(1-B(2)*B(2))):GOTO 980
WQ  960 IF B(2)=1 THEN B(3)=90:GOTO 980
UY  970 B(3)=270
MM  980 N=1:K=2:GOSUB 1290:? "↑↑[X]STOPIEN PIERWIASTKA N=";C:?
NK  990 FOR K=0 TO C-1:A(K+4)=R*COS((A(3)+360*K)/C):B(K+4)=R*SIN((A(3)+360*K)/C):? "Z(";K+1;")=";
WY  1000 IF ABS(A(K+4))>=1.0E-03 THEN ? SGN(A(K+4))*(INT(1000*SGN(A(K+4))*A(K+4)+0.1))/1000;:X=3:GOTO 1020
SC  1010 IF A(K+4)<>0 THEN ? A(K+4);:X=3
RI  1020 L=K+4:GOSUB 1350:IF C<8 THEN ?
EZ  1030 X=0:NEXT K:GOTO 1100
BL  1040 REM [****************][* PODPROGRAMY *][****************]
ET  1050 REM WPROWADZANIE DANYCH-----------
IN  1060 ? ,"POSTAC: Z=A+B*i":? :FOR L=1 TO N
OJ  1070 TRAP 1070:? "A";L;"=";:INPUT A:A(L)=A
UC  1080 TRAP 1080:? "B";L;"=";:INPUT B:B(L)=B
TN  1090 NEXT L:TRAP 40000:RETURN
CO  1100 REM POWROT DO MENU-----------------
DA  1110 ? "↓[ESC] - POWROT DO MENU";:POKE 764,255
DO  1120 IF PEEK(764)=28 THEN 120
OL  1130 GOTO 1120
YC  1140 REM MODUL-------------------------
FN  1150 MOD=(A(1)*A(1)+B(1)*B(1)):IF MOD=(SQR(MOD))^2 THEN MOD2=0:MOD=SQR(MOD):RETURN
AI  1160 MOD2=MOD:MOD=SQR(MOD):RETURN
KZ  1170 REM LOGARYTM----------------------
UV  1180 GOSUB 1140:A(N)=LOG(MOD):IF A(1)=0 THEN B(N)=1.570796:RETURN
PB  1190 RAD :B(N)=ATN(B(1)/A(1)):RETURN
JS  1200 REM EXPONENT----------------------
VE  1210 B(N)=EXP(A(N))*SIN(B(N)):A(N)=EXP(A(N))*COS(B(N)):RETURN
BU  1220 REM MNOZENIE----------------------
XC  1230 FOR L=3 TO N:H=A(N+1):A(N+1)=A(N+1)*A(L)-B(N+1)*B(L):B(N+1)=H*B(L)+A(L)*B(N+1):NEXT L:RETURN
DR  1240 REM OBSLUGA BLEDOW----------------
AG  1250 ? "↓▸ZLY ARGUMENT Z1=0":FOR L=1 TO 800:NEXT L:GOTO 120
VJ  1260 ? "↓WYNIK Z POZA DOPUSZCZALNEGO ZAKRESU":FOR L=1 TO 800:NEXT L:GOTO 120
SF  1270 IF N<2 OR N>17 THEN ? :? "LICZBA DANYCH Z POZA ZAKRESU":FOR L=1 TO 800:NEXT L:POP :GOTO 120
BB  1280 RETURN
TJ  1290 REM DRUK ROZWIAZANIA--------------
NZ  1300 ? "K           [* * * ROZWIAZANIE * * *]":? :FOR L=1 TO N:X=0:? "Z";L;"=";:IF A(L)<>0 THEN ? A(L);:X=3
RX  1310 GOSUB 1350:NEXT L:X=0:L=N+1:POKE 82,K:? :? "↓Z=";
KE  1320 IF ABS(A(L))>=1.0E-03 THEN ? SGN(A(L))*(INT(1000*SGN(A(L))*A(L)+0.1))/1000;:X=3:GOTO 1340
FM  1330 IF A(L)<>0 THEN ? A(L);:X=3
GY  1340 GOSUB 1350:POKE 82,2:? :RETURN
KB  1350 REM WYBOR ZNAKU-------------------
PR  1360 IF ABS(B(L))>=1.0E-03 THEN B(L)=SGN(B(L))*(INT(1000*SGN(B(L))*B(L)+0.1))/1000
DY  1370 IF ABS(B(L))<>1 THEN ON SGN(B(L))+2+X GOSUB 1430,1410,1420,1430,1440,1450:RETURN
PJ  1380 IF B(L)=-1 THEN ? "-i":RETURN
PR  1390 IF X=3 THEN ? "+i":RETURN
KQ  1400 ? "i":RETURN
RA  1410 ? "0":RETURN
RI  1420 ? "i*";B(L):RETURN
VA  1430 ? "-i*";-B(L):RETURN
TD  1440 ? :RETURN
TO  1450 ? "+i*";B(L):RETURN
```

SAMOURUCHAMIANIE PRZEZ ENTER

Na zakończenie toczącej się od pewnego czasu na łamach „Bajtka" dyskusji o automatycznym uruchamianiu programów w Basicu prezentujemy jeszcze jeden sposób uzyskania tego efektu.

Wykorzystane tu zostało specyficzne działanie instrukcji LIST i ENTER. Pierwsza z nich zapisuje program w postaci kodów ASCII (tak jak na ekranie). Podczas odczytu zapisanego w ten sposób programu przy pomocy instrukcji ENTER kolejne wiersze programu są umieszczone w pamięci dokładnie tak samo, jak podczas wprowadzania z klawiatury. Jeżeli do programu zostanie dodany wiersz bez numeru, to zostanie on wykonany natychmiast po odczytaniu.

Gdy w tym wierszu znajdzie się instrukcja RUN, to program zostanie automatycznie uruchomiony. Oczywiście instrukcja RUN musi się w takim przypadku znaleźć na końcu wczytywanego programu.

Normalnie podczas zapisu na taśmie nie można jednak dodać wiersza bez numeru. Czynność ta jest wykonywana przez pierwszy z zamieszczonych programów. Odczytuje on dowolny inny program nagrany na taśmie przez LIST, a następnie zapisuje go ponownie na taśmie z dodaną na końcu instrukcją RUN. Poddany tej operacji program uruchamia się samoczynnie po odczytaniu go intrukcją ENTER.

Drugi program ma podobne działanie z tą różnicą, że program do nagrania jest wprowadzany z klawiatury. Pozwala to na tworzenie krótkich programów ładujących tzw. „loaderów". Trzeba jednak uważać, aby wprowadzane wiersze były bezbłędne, ponieważ program nie kontroluje składni wprowadzanych instrukcji.

Zastosowany sposób ma jedną zasadniczą wadę. Programy zapisywane przez LIST mają długie przerwy między rekordami i w związku z tym długi czas odczytu. Praktyczne jego zastosowanie ograniczone jest więc do programów o stosunkowo niewielkiej długości, np. właśnie loaderów.

```
CV  10 CLR :GRAPHICS 0:DIM A$(6):A$="h@+LV
    @":AL=4096:IO=848
BG  20 POSITION 0,0:? "AUTOSTART-ENTER by
    [illegible]"
MQ  30 POSITION 8,5:? "[illegible]";
EP  40 OPEN #1,4,8,"C:":POKE IO+2,7
RV  50 POKE IO+4,0:POKE IO+5,16
VA  60 POKE IO+8,0:POKE IO+9,140
MF  70 X=USR(ADR(A$)):CLOSE #1:L=PEEK(IO+8
    )+256*PEEK(IO+9)
IA  75 AD=AL+L:POKE AD,82:POKE AD+1,85:POK
    E AD+2,78:POKE AD+3,155:L=L+4:REM DOPI
    SANIE [illegible] CHR$(155)
LH  80 POSITION 0,10:? "Zapis...";
IA  85 A1=INT(L/256):A0=L-A1*256
MU  90 OPEN #1,8,0,"C:":POKE IO+2,11
CR  100 POKE IO+4,0:POKE IO+5,16
MB  110 POKE IO+8,A0:POKE IO+9,A1
TP  120 X=USR(ADR(A$)):CLOSE #1
KM  130 GRAPHICS 0:END
```

```
LI  1 REM *                         *
              ZMIENNE
         *                          *
RE  10 CLR :DIM A$(2000),S$(128),C$(500):X
    =1
MX  20 OPEN #1,4,0,"E:":POKE 82,0:POSITION
     0,0
SJ  21 REM *                        *
            [illegible] OPIS
         *                          *
US  30 ? "[illegible] AUTOSTART ENTER [illegible]
    [illegible] by [illegible] Software
    [illegible]";
EF  40 ? "[illegible]
    [illegible]"
AX  41 REM *                        *
            KROTKI OPIS
         *                          *
RB  50 ? "[illegible]Wprowadz linie programu."
MX  60 ? "[illegible]Gdy chcesz zgrac tekst,w post
    aci Pliku +AUTORUN,nacisnij RETURN w p
    ustej linii.[illegible]";
OE  61 REM *                        *
           WCZYTYWANIE LINII
         *                          *
TA  70 POKE 703,4:? "K"
OB  80 INPUT #1,S$:CLOSE #1:IF S$="" THEN
    500
AY  90 A$(X,X+LEN(S$))=S$:X=X+LEN(S$)+1:A$
    (X-1,X-1)=CHR$(155)
PM  100 GOTO 20
OK  499 REM *                       *
            ZAPIS NA TASMIE
         *                          *
SE  500 POKE 566,158:OPEN #1,8,0,"C:":? #1
    ,A$;"RUN":CLOSE #1:POKE 566,146:RUN
```

Roman Tomaszewski
Andrzej Zalewski

PRZESZUKIWANIE PAMIĘCI

Podczas pisania programów w BASIC-u, w których wykonujemy operacje na zmiennych tekstowych, zachodzi czasami konieczność wyszukiwania w nich pewnych podciągów znakowych.

Również w Basicowych monitorach pamięci pomocną opcją może być możliwość wyszukiwania umieszczonych w pamięci mikrokomputera sekwencji bajtów. Napisanie procedury, która by realizowała powyższe czynności w języku Basic jest czynnością prostą, ale czas jej wykonywania zwłaszcza przy przeszukiwaniu dużych obszarów pamięci jest zbyt długi, co pogarsza efektywność pisanych programów. W związku z tym proponuję wykorzystanie do tych celów zamieszczonego poniżej programu.

Program ten zawiera w liniach 80—110 dane do relokowanej procedury maszynowej, które dla wygody użytkownika umieszczane są przez program w zmiennej tekstowej A$. Po bezbłędnym wpisaniu programu i jego uruchomieniu (dla bezpieczeństwa zapisać program na taśmie lub dyskietce) tworzona jest zmienna tekstowa A$ zawierająca naszą procedurę w liniach 10 i 20, pozostałe linie programu są kasowane. Po zapisaniu tych linii programu na nośniku zewnętrznym instrukcją LIST, możemy za pomocą instrukcji ENTER dołączać je do własnych programów.

Wywoływanie procedury:

1. W przypadku wyszukiwania sekwencji bajtów bezpośrednio w pamięci mikrokomputera postać wywołania jest następująca:

X=USR(ADR(A$),START,KONIEC, ADR(B$),LEN(B$))

gdzie B$ jest zmienną tekstową, w którą należy wpisać wcześniej wyszukiwaną sekwencję bajtów w znakach ATASCII (maksymalna długość 255 znaków);

START, KONIEC — są to adresy odpowiednio początku i końca przeszukiwanego przez procedurę obszaru pamięci.

Pod zmienną X procedura podstawia adres pierwszego bajtu odnalezionej w pamięci sekwencji. W przypadku jej nieodnalezienia podstawiana jest pod X wartość zero. Uwaga! Wartość zero podstawiana jest pod X także, gdy sekwencja rozpoczyna się od adresu 0000.

2. W przypadku wyszukiwania sekwencji bajtów w zmiennej tekstowej postać wywołania i znalezienia początku jej położenia jest nieco bardziej złożone. Jako adres startu wyszukiwania (START) podstawiamy adres przeszukiwanej tablicy np. dla tablicy S$ podstawiamy ADR(S$), adresem końca zaś w tym przypadku będzie ADR(S$)+LEN(S$)−1. Postać wywołania procedury może być więc następująca:

X=USR(ADR(A$),ADR(S$),ADR(S$)+LEN(S$)−1,ADR(B$),LEN(B$))

Po tym wywołaniu należy jako pierwszą wykonać następującą instrukcję:

X=X−ADR(S$)+1

Jeśli X≤0 to przeszukiwana tablica nie zawiera danej sekwencji, gdy zaś X>0 to wartość X wskazuje położenie szukanej sekwencji w zmiennej S$.

Na zakończenie chcę powiedzieć, że czas przeszukiwania przez procedurę obszaru pamięci 64KB wynosi nieco ponad 2 sekundy, zaś w przypadku przeszukiwania zmiennych tekstowych procedura działa szybciej niż instrukcja FIND w Basicu XL.

```
AS  1 REM PRZESZUKIWANIE PAMIECI
GF  2 REM Krzysztof Kolodziej
XN  3 REM Copyright 1988 Bajtek
SL  10 DIM A$(86),B$(255)
SD  20 ? CHR$(125):POSITION 2,4:? "20 A$="
    ;CHR$(34);
CT  30 FOR I=1 TO 86:READ X:Z=Z+X:? CHR$(X
    );:NEXT I:?
JV  40 IF Z<>13403 THEN ? :? "BLAD W DANYC
    H":END
HP  50 FOR I=30 TO 110 STEP 10:? I:NEXT I
QF  60 ? "POKE 842,12:? CHR$(125):LIST"
MI  70 POSITION 2,0:POKE 842,13:STOP
GL  80 DATA 104,104,133,204,104,133,203,10
    4,133,206,104,133,205,104,133,208,104,
    133,207,104,104,240,47,133,209
QO  90 DATA 162,0,161,203,193,207,208,14,1
    60,0,200,152,197,209,240,36,177,203,20
    9,207,240,244,165,203,197,205
AW  100 DATA 240,2,208,6,165,204,197,206,2
    40,9,230,203,208,2,230,204,24,144,213,
    169,0,133,212,133,213,96,165
LP  110 DATA 203,133,212,165,204,133,213,9
    6
```

Krzysztof Kołodziej

PEEK I POKE W KYAN PASCALU

Kyan Pascal, bardzo wygodny kompilator na małe Atari, nie posiada standardowych podprogramów PEEK oraz POKE. W instrukcjach dotyczących tego języka informuje się, że z pomocą standardowej procedury „assign" można zrealizować własne podprogramy PEEK oraz POKE.

Podane tam sposoby realizacji nie prowadzą do zadawalających rezultatów, najprawdopodobniej z powodu niezrozumienia istoty procedury „assign". Poniżej przedstawione propozycje PEEK i POKE zrealizowano w Kyan Pascalu wersja 1.0 (ED wersja 1.1, PC wersja 1.2.).

Procedura „assign" pozwala umieszczać w dowolnym miejscu pamięci wartości wcześniej zadeklarowanych zmiennych. Wywołanie ma postać: assign (wskaźnik, miejsce). „Wskaźnik" jest zmienną wskazującą na zadeklarowaną wcześniej inną zmienną, „miejsce" jest typu integer i równa się numerowi komórki, na którą wskazywać będzie „wskaźnik" po wykonaniu procedury. Dokładniej, wskaźnik wskazywać będzie na ciąg komórek, począwszy od komórki o numerze „miejsce". Długość tego ciągu zależy od wcześniejszej deklaracji na jaki typ ma wskazywać „wskaźnik".

Zamieszczony poniżej program „pokeReal" umieszcza w pamięci począwszy od komórki o numerze „początek"=12100 cztery liczby typu real, wczytywane z klawiatury. Ponieważ umieszczane będą zmienne typu real „wskaźnik" zadeklarowano jako „wskazuje na real". Dokonano tego pośrednio poprzez zdefiniowanie typu „adres". Właściwe wstawienie kolejnej liczby następuje podczas realizacji instrukcji podstawienia „wskaźnik^:=liczba1", którą należy odczytywać: w osiem kolejnych bajtów, począwszy od komórki o numerze „początek+8xi" wstaw liczbę „liczba1". Zmienna „i" oprócz sterowania pętlą FOR przesuwa „wskaźnik" co osiem komórek po wstawieniu kolejnej liczby. Druga pętla FOR odczytuje wstawione liczby, ale w kolejności odwrotnej. Przedstawiony program ma charakter poglądowy. Bardziej praktyczne są dwa następne: procedura „poke" oraz funkcja „peek".

Ponieważ oba podprogramy operują na pojedynczej komórce pamięci „wskaźnik" powinien być typu „wskazuje na zmienną zajmującą jedną komórkę". W realizacji wybrano typ „char". Ma on ponadto tę zaletę, że z pomocą funkcji standardowych „chr" oraz „ord" łatwo przeprowadzić go w typ integer i odwrotnie. Ze zmienną „miejsce" wiąże się sposób adresowania pamięci stosowany w procedurze „assign". Należy pamiętać ze MAXINT=32767. Komórki powyżej tego numeru są adresowane liczbami ujemnymi.

W zamieszczonych programach „peek" oraz „poke" przyjęto, ze adres komórki określany będzie zmienną „gdzie" typu real. Wewnątrz podprogramu zmienna ta będzie podlegała konwersji na zmienną „miejsce" typu integer. Następnie procedura „assign" ustawi „wskaźnik" nad właściwą komórką. Końcowe instrukcje wstawią do komórki znak o rządanym kodzie lub odczytają go i zmienią na kod.

Ponieważ kompilator nie przeprowadza koniecznych konwersji typów przy wywoływaniu procedur i funkcji niestandardowych, omawiane podprogramy należy wywoływać z parametrem „gdzie" typu real. Tak na przykład peek (20.0), poke (77.0,128), ale peek (53279), gdyż ze względu na wielkość argument jest już typu real.

Należy zaznaczyć, ze omawiana procedura „assign" nie ma nic wspólnego — poza nazwą — z podobną procedurą w Turbo-Pascalu. W Kyan Pascalu utożsamianie pliku z konkretnym zbiorem na dysku realizowane jest przez odpowiednią formę procedur „reset" oraz „rewrite", na przykład: „rewrite" (g, 'B1: NOWYPLIK')".

Ryszard Wiech

```
PROGRAM pokeReal(input,output);
 TYPE adres=^real;
 VAR  wskaznik:adres;
      liczba1,liczba2:real;
      poczatek,i:integer;
 BEGIN
   poczatek:=12100;
   FOR i:=0 TO 3 DO
     BEGIN
       readln(liczba1);
       assign(wskaznik,poczatek+8*i);
       wskaznik^:=liczba1
     END;
   FOR i:=3 DOWNTO 0 DO
     BEGIN
       assign(wskaznik,poczatek+8*i);
       liczba2:=wskaznik^;
       writeln(liczba2:8:4)
     END
 END.


FUNCTION peek(gdzie:real):integer;
 TYPE bajt=char;
      adres=^bajt;
 VAR  wskaznik:adres;
      miejsce:integer;
 BEGIN
   IF gdzie>32767
     THEN
       miejsce:=trunc(gdzie-65536)
     ELSE
       miejsce:=trunc(gdzie);
   assign(wskaznik,miejsce);
   peek:=ord(wskaznik^)
 END;


PROCEDURE poke(gdzie:real;co:integer);
 TYPE bajt=char;
      adres=^bajt;
 VAR  wskaznik:adres;
      miejsce:integer;
 BEGIN
   IF gdzie>32767
     THEN
       miejsce:=trunc(gdzie-65536)
     ELSE
       miejsce:=trunc(gdzie);
   assign(wskaznik,miejsce);
   wskaznik^:=chr(co)
 END;
```

ZOSTAŃ NIEŚMIERTELNYM! (6)

Tomasz Hryszko, uczeń I klasy LO z Białegostoku przysłał nam wspaniały prezent świąteczny — 7 ułatwień do gier.

MR. DO
Nieograniczona liczba „żyć" da nam zamiana rozkazu DEC $0618,X na LDA $0618, X ($DE, $18, $06 na $BD, $18, $06 — "SHIFT-*" w negatywie, "CTRL-X", "CTRL-F" na "=" w negatywie, "CTRL-X", "CTRL-F")

PIE MAN
Liczba „żyć" jest przechowywana w komórce $80. Należy zamienić DEC $80 na LDA $80 ($C6, $80 na $A5, $80 — "F" w negatywie, "CTRL-," w negatywie na "%" w negatywie, "CTRL-," w negatywie).

HYPERBLAST
Liczba „żyć" w komórce $9A — zamienić DEC $9A na LDA $9A ($C6, $9A na $A5, $9A — "F" w negatywie, "CTRL-Z" w negatywie na "%" w negatywie, "CTRL-Z" w negatywie).

PHOBOS
Liczba „żyć" w komórce $1285 — zamienić DEC $1285 na LDA $1285 ($CE, $85, $12 na $AD, $85, $12 — "N" w negatywie, "CTRL-E" w negatywie, "CTRL-R" na "—" w negatywie, "CTRL-E" w negatywie, "CTRL-R")

CAVERNS OF MARS II
Liczba „żyć" w komórce $3B39 — zamienić DEC $3B39 na LDA $3B39 ($CE, $39, $3B na $AD, $39, $3B — "N" w negatywie, "9", ";" na "—" w negatywie, "9", ";").

ASTEROIDS
Liczba „żyć" w komórce $B1 — zamienić DEC $B1,X na LDA $B1,X ($D6, $B1 na $B5, $B1 — "V" w negatywie, "1" w negatywie na "5" w negatywie, "1" w negatywie).

DIG-DUG
Liczba „żyć" w komórce $1805 — zamienić DEC $1805,X na LDA $1805,X ($DE, $05, $18 na $BD, $05, $18 — "SHIFT-*" w negatywie, "CTRL-E", CTRL-X" na "=" w negatywie, "CTRL-E", "CTRL-X").

Ireneusz Piotrzkowicz z Otwocka pomógł nam poprawić grę: SWAT
Należy zamienić DEC $066B na LDA $066B ($Ce, $6B, $06 na $AD, $6B, $06 — "N" w negatywie, "k", "CTRL-F", na "—" w negatywie, "k", "CTRL-F").

Bartosz Smaga z Warszawy przysłał poprawkę do gry: DAN STRIKES BACK
Zamieniamy rozkaz DEC $0608 na LDA $0608 ($DE, $08, $06 na $AD, $08, $06 — "N" w negatywie, "CTRL-H", "CTRL-F" na "—" w negatywie, "CTRL-H", "CTRL-F").

Od 13-letniego Mariusz Geborka z Dankowic mamy poprawkę do gry:
CAPTAIN STICKY'S GOLD
Odszukujemy rozkaz DEC $064B i likwidujemy go przez wpisanie w jego miejsce 3 razy NOP ($CE, $4B, $06 na 3 razy $EA — "N" w negatywie, "K", "CTRL-F" na 3 razy "j" w negatywie).

A teraz kilka poprawek, które sam odkryłem:

THE GOONIES
Zamieniamy DEC $112C na LDA $112C ($CE, $2C, $11 na $AD, $2C, $11 — "N" w negatywie, ",", "CTRL-Q" na "—" w negatywie, ",", "CTRL-Q").
Rozkaz ten występuje w programie dwukrotnie, trzeba więc dokonać zamiany dwa razy

MARIO BROS
Liczba „żyć" w komórce $29 — zamieniamy DEC $29 na LDA $29 ($C6, $29 na $A5, $29 — "F" w negatywie, ")" na "%" w negatywie, ")").

THE LIVING DAYLIGHTS
Odporność na strzały wroga da nam zamiana DEC $80AF na LDA $80AF ($CE, $AF, $80 na $AD, $AF, $80 — "N" w negatywie, "/" w negatywie, "CTRL-," w negatywie na "—" w negatywie, "/" w negatywie, "CTRL-," w negatywie).

Czekamy na listy z poprawkami od czytelników — bardzo chętnie je opublikujemy.

Tomasz Wiśniewski

MONITORY ML (1)

O programy te pyta nas coraz więcej Czytelników. Do czego służą? Jak je wykorzystać? Co można dzięki nim uzyskać? — oto pytania na które postaram się odpowiedzieć w tej serii artykułów.

W języku angielskim słowo „monitor" oznacza nie tylko monitor telewizyjny, ale również podgląd czy wgląd. W rzeczywistości komputerowej są to programy (lub instrukcje BASIC) umożliwiające nam programowanie w języku wewnętrznym, dekodowanie takich programów oraz ich badanie, usuwanie błędów itp. Zestaw instrukcji jakie obejmuje zwykły monitor języka maszynowego czyni go uniwersalnym narzędziem do pracy nad programami tworzonymi w tym języku. W dużych pakietach (np. CBM MACRO ASSEMBLER DEVELOPMENT SYSTEM) spełniają one rolę pomocniczą, gdyż np. układanie programu jest znacznie prostsze dzięki dużym asemblerom, pracującym w oparciu o etykiety (do tego tematu jeszcze wrócimy).

Nie ma komputera Commodore do którego nie napisano by monitora. Dla VIC-20 był to TINY MON, dla C-64 — HES MON czy monitory ze wspominanego wyżej pakietu. Rodzina C-16 ma wbudowany monitor języka maszynowego o funkcjach w zasadzie identycznych z C-64. Najlepszym monitorem, jaki do tej pory spotkałem, jest monitor C-128, w którym można znaleźć wiele bardzo przydatnych (zwłaszcza dla początkujących !) funkcji. Przykładem może tu być np. znak „+", który umożliwia wprowadzanie adresów czy argumentów w postaci dziesiętnej a nie szesnastkowej, automatyczne w zasadzie przełączanie banków itp.

O powszechności tych programów świadczy najlepiej fakt, że nawet w specjalnych kartach (modułach) typu EXPERT czy FINAL CARTRIDGE programy te są w zasadzie obowiązkowe — nie spotkałem dotychczas żadnej karty wielofunkcyjnej nie zawierającej monitora języka maszynowego, co więcej — np. EXPERT to nic innego jak potężny wielofunkcyjny monitor.

Wszystkie te programy bez względu na typ komputera są do siebie zbliżone. Poszczególne instrukcje są jednoliterowe i obejmują zwykle od kilku do kilkunastu liter alfabetu. W lepszych wersjach — jak wspominany tu EXPERT — lista instrukcji zajmuje wszystkie litery i na dodatek dużą ilość znaków typu „$" czy „#". Ponieważ jednak nie wszystkich będzie stać na zakup takiej karty, w naszych rozważaniach ograniczymy się do programów powszechnie dostępnych.

Użytkownicy C-16/116, PLUS/4 i C-128 mają taki monitor pod ręką — wystarczy wpisać MONITOR i wcisnąć klawisz RETURN. Nieco gorzej wygląda sprawa z odpowiednimi programami dla C-64, choć np. jedna z wersji SIMON'S BASIC zawiera taki monitor dostępny za pomocą —M. W „normalnym" BASIC-u jednak program tego typu trzeba po prostu wczytać do pamięci posługując się przy tym instrukcją LOAD „NAZWA",1,1 lub ,8,1. Chodzi tu o fakt, że program ten MUSI zawsze znajdować się w odpowiednim miejscu pamięci. Zwykle monitor taki uruchamia się za pomocą SYS adres, gdzie adres przyjmuje jedną z następujących wartości: 4096, 8192, 12288, 16384, 20480, 24576, 28672, 32768 lub 49152 (dla C-64). Programy te zajmują zwykle 4 KB pamięci i najwygodniejszą dla nich lokalizacją jest górny obszar pamięci RAM od adresu 49152.

Po wykonaniu odpowiedniej instrukcji SYS na ekranie ukaże się następujący komunikat:

```
    PC    SR   AC   XR   YR   SP
;  nnnn   nn   nn   nn   nn   nn
```

Litery nnnn i nn symbolizują wartości podane w kodzie szesnastkowym (heksadecymalnym). Niektóre monitory wyświetlają jeszcze dodatkowo:

```
    PC    SR   AC   XR   YR   SP    NY-BDIZC
;  nnnn   nn   nn   nn   nn   nn    01-11010
```

lub też pole o szerokości 8 do 16 znaków, w którym wyświetlane są kody CHR$ od wartości zapisanej w danej komórce.

PC — licznik programu (program counter). Jest to 16-bitowy rejestr wewnętrzny procesora, w którym znajduje się adres następnej w kolejności komórki, z której należy pobrać instrukcję do wykonania lub argument instrukcji. Jeżeli podczas uruchamiania monitora na ekranie ukaże się wartość, dajmy na to $C0FF w liczniku programu, to oznacza to, że ostatnio wykonaną instrukcją była instrukcja w komórce $C0FE. Licznik programu pozwala nam również na znacznie ciekawszą działalność — testowanie ukrytych i niepublikowanych instrukcji mikroprocesora. Dzięki analizie zawartości poszczególnych rejestrów jesteśmy w stanie określić, czy dany kod może być taką instrukcją czy też nie. Nawiasem mówiąc jest to jedyna chyba możliwość dokładniejszego poznania swego mikroprocesora.

SR — rejestr słowa stanu (status register). Informuje on o stanie logicznym poszczególnych znaczników mikroprocesora, albo w postaci wartości całego bajtu, albo też pod tajemniczo wyglądającym napisem NY-BDIZC. Zwróć uwagę, że liczba znaków w tym napisie jest taka sama jak liczba bitów w bajcie. Każdy znacznik jest w rzeczywistości reprezentowany przez określony bit tego rejestru. Znaczniki te informują procesor o wielu bardzo istotnych wydarzeniach np. czy w wyniku jakiejś operacji otrzymaliśmy wartość większą, mniejszą bądź równą zeru, czy podczas zliczania w pętli przekroczyliśmy już liczbę 255 itp. Do znaczników tych wrócimy znacznie później przy omawianiu konkretnych przykładów programów w języku wewnętrznym. Jeżeli pod określoną literą (NY-BDIZC) występuje jedynka lub „haczyk" (CHR$(186)) to oznacza to, że dany bit jest ustawiony, czyli w stanie logicznym 1.

Poszczególne litery oznaczają:

N — znacznik ujemności (Negative flag).
Y — znacznik przepełnienia (Overflow flag).
- — niewykorzystany.
B — znacznik przerwania programu (Break flag).
D — znacznik włączenia dziesiętnego trybu pracy procesora (Decimal flag).
I — znacznik wyłączenia przerwań (Interrupt disable flag).
Z — znacznik zera (Zero flag).
C — znacznik przeniesienia (Carry flag).
AC — akumulator czyli główny rejestr mikroprocesora. Wartość nn podaje jego aktualną zawartość.
XR — rejestr indeksowy X oraz jego aktualna zawartość.
YR — jw. wyżej lecz dotyczy rejestru indeksowego Y.
SP — wskaźnik stosu (stack pointer). Wskaźnik ten informuje procesor o pierwszej wolnej komórce pamięci w jej obszarze zwanym stosem mikroprocesora. Obszar ten jest bardzo intensywnie wykorzystywany przez mikroprocesor do zapamiętywania danych tymczasowych, adresów, skoków powrotnych itp. Także programista może korzystać ze stosu, co wymaga jednak dużej wprawy. Warto pamiętać, że całkowite zapełnienie stosu powoduje zwykle wyświetlenie komunikatu OUT OF MEMORY ERROR (brak pamięci) bez względu na to, czy RAM dla BASIC jest zajęta czy nie.

Zanim przejdziemy do listy instrukcji, musimy powiedzieć sobie coś niecoś o wymaganiach, jakie stawia nam monitor języka maszynowego ze swojej strony. A więc po pierwsze olbrzymia większość monitorów pracuje wyłącznie w oparciu o kod szesnastkowy, w którym po liczbie 9 występują jeszcze litery symbolizujące dalsze wartości do 16 (A=10, B=11, C=12, D=13, E=14, F=15) przy czym 16 dziesiętnie jest oznaczane jako 10 szesnastkowo. Dla odróżnienia od zapisu dziesiętnego, liczby w kodzie szesnastkowym są poprzedzane znakiem „$".

Ponieważ wiem z doświadczenia, że kod heksadecymalny nie jest specjalnie lubiany przez początkujących programistów, zamieszczam program umożliwiający przeliczanie wartości dziesiętnych na szesnastkowe i odwrotnie. Działa on na wszystkich komputerach Commodore, przy czym posiadacze C-16/116, PLUS/4 (a także C-128) mogą korzystać z funkcji HEX$ oraz DEC. W wypadku C-128 jest to w zasadzie niepotrzebne, ponieważ wystarczy liczbę dziesiętną poprzedzić znakiem „+" (np. LDA+53280) i monitor zamieni ją samoistnie na wartość szesnastkową.

Po trzecie wszelkie programy tworzone za pomocą monitora powinny być ZAWSZE najpierw zapisywane na dyskietce czy taśmie i dopiero potem uruchamiane. Język wewnętrzny obsesyjnie nie znosi niedokładności (i niedokładnych) i płata nieraz bardzo nieprzyjemne figle blokując komputer i niszcząc tym samym czasami wielogodzinny wysiłek.

Po czwarte do naszych prób będziemy potrzebowali określonego obszaru pamięci, w którym możliwe będzie swobodne i niezakłócone działanie; dla naszych celów wybrałem obszar od adresu 10000 ($2710) do 16383, co pozwoli nam na ujednolicenie wszystkich przykładów — będą one dostępne i możliwe do wprowadzenia zarówno dla użytkowników C-16 czy 116, C-64, C-128 itp. Jeżeli jednak z jakichś powodów zdecydujesz się na wpisywanie programów demonstracyjnych w innym obszarze pamięci, to możesz skorzystać z zamieszczonego w tym numerze BAJTKA artykułu PERFOROWANA PAMIĘĆ, omawiającego wolne obszary RAM w poszczególnych typach komputerów Commodore.

UWAGA. Najprostszym sposobem umożliwiającym przeliczanie adresów dziesiętnych na szesnastkowe i odwrotnie jest w C-64 wczytanie najpierw prezentowanego tu programu, a dopiero później monitora. Teraz uruchom monitor za pomocą odpowiedniego SYS. Gdy zajdzie potrzeba przeliczenia adresu, zakończ pracę z monitorem za pomocą X i RETURN, co pozwoli Ci na powrót do BASIC. Wykonaj teraz obliczenie i wróć do monitora za pomocą odpowiedniego SYS, choć może się zdarzyć i tak, że nie każdy monitor pozwoli Ci na rzeczywisty powrót do BASIC. Na przykład monitor z pakietu CBM MACRO ASSEMBLER DEVELOPMENT SYSTEM potrafi bardzo skutecznie zablokować BASIC.

W następnej części zapoznamy się bliżej z listą poleceń wspólną dla większości monitorów języka maszynowego oraz z pewnymi wyjątkami takimi jak np. instrukcje I czy J.

(cdn)

Klaudiusz Dybowski

```
8B   100 REM *** HEX-DEC & DEC-HEX ***
38   105 :
12   110 DN$=CHR$(17):PRINT CHR$(147)
35   115 INPUT "HEX-DEC (1) CZY DEC-HEX (2): ";Z
35   120 IF Z<>1 AND Z<>2 THEN RUN
1C   125 IF Z = 2 GOTO 145
10   130 PRINT:HX$="":D=0:INPUT "LICZBA SZESNASTK
         OWA: ";HX$
98   135 FOR J=1 TO LEN(HX$):M$=MID$(HX$,J,1):D=D
         *16+ASC(M$)-48+(M$>="A")*7:NEXT
6C   140 PRINT DN$;"$";HX$;" = ";D:PRINT DN$:GOTO
          115
09   145 PRINT:HX$="":D=0:INPUT "LICZBA DZIESIETN
         A: ";D:LD=D
FF   150 T=((D/16)-INT(D/16))*16:HX$=CHR$(T+48-(T
         >9)*7)+HX$:D=(D-T)/16:IF D GOTO 150
C4   155 PRINT DN$;LD;" = $";HX$:PRINT:GOTO 115
```

PERFOROWANA —PAMIĘĆ—

Wolne obszary pamięci RAM są bardzo przydatne nie tylko do wpisywania tajnych haseł czy kodów zabezpieczających program przed włamywaczem — są one najbardziej potrzebne przede wszystkim programującym w języku maszynowym. Ponieważ w tym numerze BAJTKA znajdziesz także artykuł o monitorach języka maszynowego informacja o tym gdzie można wpisywać własne programy w tym języku może okazać się bardzo przydatne.

COMMODORE 64:

— 2 oraz komórki 251 — 254 na stronie zerowej.
— 679 — 767 (wykorzystywany czasami do przechowywania danych o sprite'ach).
— 784 — 786 pod warunkiem, że w programie nie wykorzystujemy instrukcji USR().
— 814 i 815 — przeznaczone na wektor procedury użytkownika.
— 820 — 827.
— 828 do 1019 włącznie. Jest on zarezerwowany dla bufora kasety stąd też należy pamiętać, że program lub dane w nim zawarte zostaną skasowane podczas jakiejkolwiek operacji z magnetofonem.
— 1020 do 1023.
— 2040 — 2047 pod warunkiem, że nie korzystamy w programie ze sprite'ów.
— 2049 do 40959 czyli normalna pamięć RAM dla programu i zmiennych.
— 49152 — 53247 czyli obszar dodatkowych 4 KB nieadresowalnych przez BASIC.
— 53248 — 53264, 53271 jeżeli nie korzystamy ze sprite'ów.
— 53275 — 53279 jak wyżej.
— 5328? — 53294 jak wyżej.

COMMODORE 128:

— 250 — 254 pięć komórek na stronie zerowej.
— 852 — 861 jeśli nie będziemy korzystać z klawisza TAB.
— 1021 — 1023.
— 2810 — 3071 obszar dla bufora kasety; jakakolwiek operacja z magnetofonem spowoduje skasowanie wpisanego tu programu lub danych.
— 3072 — 3583 jeśli nie będziemy korzystali z RS-23?.
— 3584 — 4095 jeśli nie korzystamy ze sprite'ów.
— 4096 — 4351 jeśli nie korzystamy z klawiszy funkcyjnych.
— 4632 — 4634 jeśli nie ma instrukcji USR() w programie.
— 4864 — 7167.
— 7169 — 65535 czyli zwykła pamięć RAM dla użytkownika (tylko program) jeśli wykonamy najpierw BANK 0.
— 0000 — 65535 jeżeli korzystamy wyłącznie z języka wewnętrznego i banku 1 (przeznaczonego na zmienne programu)

COMMODORE 16/116 I PLUS/4:

— 208 — 232.
— 216 — 232.
— 818 — 1010 czyli bufor kasety — dane lub program zostanie stąd usunięty jeżeli będziemy wczytywać lub zapisywać cokolwiek na taśmie.
— 1015 — 1078 jeżeli nie korzystasz z RS-232 (dotyczy tylko PLUS/4).
— 1280 — 1282 jeżeli nie ma w programie instrukcji USR ().
— 1373 — 1510 jeśli nie korzystasz z klawiszy funkcyjnych.
— 4096 — 16383 czyli zwykła pamięć RAM dla użytkownika (dla C-16/116 16 KB).
— 4096 — 64767 czyli zwykła pamięć RAM dla użytkownika (C-16/116 64 KB, PLUS/4).

Oczywiście takich obszarów możnaby znaleźć więcej gdyż zależy to w dużej mierze od zadania jakie ma wykonać program. I tak jeśli nie używamy stacji dysków to do naszej dyspozycji stoi również obszar przeznaczony na potrzeby dyskowego systemu operacyjnego (DOS), jeżeli niepotrzebna jest nam grafika — możemy korzystać z większości komórek dla niej przeznaczonych itp. Zwróć jednak uwagę, że wszystkie komputery mają dwa duże obszary wspólne: 4864 — 7167 oraz 7169 — 16383. Jeżeli więc Twój program w języku wewnętrznym nie wykorzystuje specyficznych własności układów np. VIC czy TED, wektorów systemu operacyjnego itp., to będzie on działał na wszystkich typach komputerów jeżeli zostanie umieszczony w jednym z tych obszarów.

Do czego jeszcze obszary te czy nawet pojedyncze komórki mogą być przydatne? Zastosowań jest wiele. Możesz za pomocą POKE w BASIC wpisywać do nich np. wartości kontrolowane następnie przez program; jeżeli nie są one zgodne z wartościami przypisanymi im przez Ciebie program np. automatycznie się kasuje czy formatuje dyskietkę. Jeżeli teraz wartości te nie będą na stałe zapisane w programie lecz będziesz je wprowadzał za pomocą INPUT i porównywał z wartościami wpisanymi tam PRZED wczytaniem programu to zabezpieczenie to będzie niezmiernie trudne o ile wogóle możliwe do złamania przez pirata. W kilkunasto- czy kilkudziesięciobajtowych obszarach RAM można umieszczać własne krótkie programy czy procedury w języku wewnętrznym uzupełniające program główny, zapisywać dane dotyczące dźwięku czy stałych wartości (zamiast marnować RAM i przypisywać je zmiennym) itp.

Klaudiusz Dybowski

OBWOLUTA

```
6C  100 REM   *** OBWOLUTA  ***
3B  105 :
AF  110 REM   *** K. DYBOWSKI ***
F8  115 :
78  120 :
3C  125 DIM A$(26),S$(13),C$(26)
5D  130 PRINT CHR$(147):D$=CHR$(17)
AA  135 R$="|
                    |"
1A  140 FOR I=1 TO 13:READ Z$:S$(I)=Z$:NEXT
87  145 INPUT "NR KASETY :";N$:IF N$="" GOT
        O 145
46  150 IF LEN(N$)=1 THEN N$=N$+"   ":GOTO
        165
EF  155 IF LEN(N$)=2 THEN N$=N$+"  ":GOTO 1
        65
63  160 N$=N$+" "
F6  165 PRINT D$"STRONA A"D$
F8  170 FOR I=1 TO 13
82  175 PRINT "NAZWA PROGRAMU";I;:INPUT A$(
        I)
0B  180 IF LEN(A$(I))>13 GOTO 175
BE  185 GOSUB 350:NEXT:PRINT CHR$(147)
CE  190 PRINT "STRONA B"D$
11  195 FOR I=14 TO 26
A0  200 PRINT "NAZWA PROGRAMU";I-13;:INPUT
        A$(I)
C9  205 IF LEN(A$(I))>13 GOTO 200
E9  210 GOSUB 350:NEXT
0F  215 :
43  220 CLOSE4:OPEN4,4
28  225 PRINT#4,"┌──────────────────────────
        ───────────┐"
85  230 PRINT#4,"| KASETA NR :";N$;"
                    |"
15  235 PRINT#4,R$
10  240 :
AF  245 FOR I=1 TO 13
02  250 PRINT#4,"|";C$(I);A$(I);C$(I);A$(I+
        13);"|"
E2  255 NEXT
16  260 PRINT#4,"├──────────────────────────
        ───────────┤"
66  265 PRINT#4,"| STR A:
                    |"
37  270 PRINT#4,"| STR B:
                    |"
18  275 PRINT#4,"├──────────────────────────
        ───────────┤"
A8  280 FOR X=1 TO 5:PRINT#4,R$:NEXT
BC  285 PRINT#4,"└──────────────────────────
        ───────────┘"
57  290 PRINT#4:CLOSE4
15  295 :
F8  300 PRINT D$:INPUT "KONIEC (T/N) ";KN$
03  305 IF KN$<>"T" THEN RUN
BB  310 END
45  315 :
E2  320 DATA " ","  ","   ","    ","     "
C1  325 DATA "      ","       ","        "
1C  330 DATA "         ","          "
4D  335 DATA "           ","            "
8C  340 DATA "             "
87  345 :
BB  350 D=LEN(A$(I)):U=(13-D)
FA  355 IF D<13 THEN A$(I)=A$(I)+S$(U)
87  360 IF I<=13 THEN SD$="    ":GOTO 370
95  365 SD$=" "
3B  370 A$(I)=A$(I)+SD$:IF I<10 THEN C$(I)=
        " "+STR$(I)+" ":RETURN
E   375 C$(I)=STR$(I)+" ":RETURN
```

Dwa poniżej przedstawione programy pozwalają na samodzielne wykonanie koperty dla dyskietki oraz papierowej wkładki do pudełka od kasety. Programy te współpracują z dowolnym Commodore i ze wszystkimi drukarkami, które mogą współpracować z tymi komputerami. W wypadku wkładki do pudełka od kasety możliwa jest także aktualizacja danych dotyczących programów zawartych na danej kasecie. Możliwe jest wpisanie do 13 tytułów programów z obu stron kasety. Gdy po jakimś czasie taśma ta znudzi Ci się i skasujesz zawarte na niej programy, to bez większych problemów możesz zrobić dla niej nową wkładkę zawierającą aktualne tytuły programów. W wypadku programu drukującego kopertę dla dyskietki użytkownik powinien w miejsce „Imię i nazwisko" wpisać swoje własne dane nie zmieniając przy tym liczby znaków w tym wierszu. Może to być np. imię i nazwisko właściciela, dowolny znak katalogowy itp. Papier, z którego zostanie wykonana koperta, powinien być w miarę sztywny i czysty — doskonale do tego celu nadaje się np. papier kredowy.

Jeżeli Twoja drukarka współpracuje z komputerem poprzez port użytkownika, to musisz najpierw wprowadzić do programu odpowiednie korekty (dotyczy to instrukcji OPEN oraz PRINT#4). Oczywiście posiadacze lepszych drukarek np. NL-10 czy GEMINI będą mogli z łatwością wprowadzić poprawki do programów, umożliwiające wydruk więcej aniżeli 13 tytułów w wypadku wkładki czy np. dodatkowe pola dla nazwy i identyfikatora dyskietki. Oprócz programów zamieszczam także przykładową wkładkę, jaką otrzymamy w wyniku działania pierwszego programu.

Klaudiusz Dybowski

```
6b  100 l$ ="┌──────────────────────────
        ──────────────────┐"
67  105 d$="|        |
                          |       |"
a2  110 d1$="|       | Imie i nazwisko
        Dyskietka nr: ______ |       |"
18  115 d2$="        |
                          |"
95  120 l2$="        └──────────────────
        ──────────────────┘"
8a  125 open 4,4,7:print#4,l$:print#4,d$:print
        #4,d1$
d5  130 for i=1 to 21:print#4,d$:next
f8  135 print#4,l$
79  140 for i=1 to 28:print#4,d2$:next
a5  145 print#4,l2$:print#4:close 4:end
```

```
KASETA NR :155

  1 STAR TREK 3          1 ZAXXON 3
  2 ZOIDS                2 IMP. MISSION
  3 SOLO FLIGHT 2        3 DESERT FOX
  4 SPITFIRE '40         4 WIZBALL
  5 OGRE                 5 BATTLE SHIPS
  6 SOLDIER 2            6 ENTOMBED
  7 ALTER EGO            7 STARDUST
  8 STRIKE FLEET         8 JACKAL
  9 F. WARRIOR           9 TETRIS
 10 STEALTH 2           10 EAGLE'S NEST
 11 PEACEKEEPER         11 FIGHTER PILOT
 12 MIND SHADOW         12 SOLDIER 3
 13 ELITE               13 TRAZ
---------------------------------------
STR A:
STR B:
---------------------------------------
```

PRZEDSTAWIAMY WARSAW BASIC (11)

PRZEKAZYWANIE WYNIKÓW Z PROCEDURY

Często zdarza się, że wynikiem działania procedury ma być jakaś wartość numeryczna lub tekstowa.

Jeśli do obliczenia tej wartości nie trzeba posługiwać się skomplikowanym wzorem, to aby zaprogramować taką procedurę można zastosować deklarację DEF FN. Jeśli program, który przypisuje FN jakąś wartość musiałby mieć więcej niż jeden wiersz, to DEF FN już nie wystarczy.

Aby pomóc w uporaniu się z podobnymi problemami w Warsaw BASIC'u zastosowano mechanizm przekazywania parametrów przez nazwę do i z procedury. W tym przypadku parametr aktualny w wywołaniu procedury może być zmienną prostą, nazwą tablicy lub nazwą funkcji. W czasie wykonywania instrukcji PROCEDURE ta aktualna nazwa parametru jest podstawiana zamiast odpowiedniego parametru formalnego. Zmiana wartości takiego parametru formalnego w procedurze powoduje modyfikację odpowiadającego mu parametru aktualnego w programie wywołującym. Na czas działania procedury parametr aktualny i formalny mają wspólne pole wartości, choć mogą mieć zupełnie różne nazwy. Przekazywanie przez nazwę jest jedynym sposobem przesyłania używanym w przypadku tablic i funkcji. W przypadku zmiennych prostych stosowane bywa właśnie wtedy, gdy dany parametr reprezentuje wynik działania procedury. W Warsaw BASIC'u procedura może mieć dość dowolną liczbę takich wyjść zarówno numerycznych jak i tekstowych.

W implementacjach innych języków, np. Logo, można się spotkać z innym rozwiązaniem. Wynik działania procedury jest przesyłany do segmentu wywołującego przez jej nazwę. Tym samym procedura jest funkcją swoich parametrów. Ogranicza to liczbę wyjść z procedury do jednego, ale bywa

uzyteczne przy rekursyjnym wywoływaniu procedur. Ten dogodny sposób programowania procedur ma też swoje wady podobne do tych jakie mają funkcje zdefiniowane za pomocą DEF FN. Otóż wszystkie zmienne poza parametrami formalnymi są w niektórych implementacjach Logo zmiennymi globalnymi. Tak więc wywołanie takiej procedury w innym środowisku niz to w jakim została stworzona zawsze wymaga wyeliminowania powtarzających się przypadkowo nazw zmiennych globalnych. Jak wiemy Warsaw BASIC jest pozbawiony tej niedogodności.

W interpreterze rozbudowanym na użytek Czytelników Bajtka zastosujemy rozwiązanie podobne do tego jakie występuje w interpreterach Logo. Jedna z różnic to brak możliwości rekursyjnego wywoływania w naszym interpreterze, co spowodowane jest mechanizmem umieszczania procedury w pamięci za kazdym razem, gdy jest ona wołana. Nie ma tego ograniczenia w wersji WB 3.2, gdzie po umieszczeniu procedury w RAM-dysku mozna stosować rekursję ograniczoną tylko pojemnością stosu. Oczywiście zastosowanie w naszym interpreterze sposobu przesyłania podobnego do funkcji FN nie niesie za sobą tych problemów o jakich była mowa w przypadku procedur Logo.

W interpreterze rozbudowanym dla czytelników Bajtka wywołanie procedury jako funkcji moze mieć postać:

X=£C"ZOSIA".8

w przypadku wywołania z dyskietki lub:

X=£C"ZOSIA"

w przypadku wywołania z taśmy magnetycznej.

W przypadku takiego wywołania procedury rozbudowany interpreter przypisze zmiennej X w programie wywołującym wynik ostatniej operacji arytmetycznej wykonanej w procedurze przed wypełnieniem rozkazu £E. Jeśli zatem ostatnią operacją będzie np. instrukcja podstawienia:

A=B+5/C,

to rozbudowany interpreter przypisze zmiennej X wartość zmiennej A po wykonaniu działań B+5/C. Mozna zatem powiedzieć, ze dokona tego co interpreter wersji WB 3.2 w sytuacji, w której X byłby parametrem aktualnym a A odpowiadającym mu parametrem formalnym przesyłanym przez nazwę. Przy pomocy rozbudowanego dla Czytelników Bajtka interpretera można przekazywać w ten sposób tylko dane numeryczne.

Opisane powyżej funkcje realizuje program 1. Jeśli chcą je zrealizować Czytelnicy Bajtka, to powinni dołączyć program 2 do poprzednich części rozbudowanego interpretera i uruchomić całość. Życzymy powodzenia!

Do tej pory mieliśmy do czynienia z jednym rodzajem zmiennych — wszystkie zmienne, czy to w programie wywołującym, czy to w procedurach były zmiennymi lokalnymi. Zmienne lokalne mogą mieć te same nazwy, ale zawsze mają różne wartości. Trochę inny rodzaj zmiennych, to parametry formalne przekazywane przez nazwę. Taki parametr formalny i odpowiadający mu parametr aktualny mogą mieć różne nazwy ale mają takie same wartości. W Warsaw BASIC'u występuje jeszcze trzeci rodzaj zmiennych, tzw. zmienne globalne. Zmienne globalne wszędzie (w kazdym segmencie) muszą się tak samo nazywać i mają takie same wartości. O tym jak zrealizowaliśmy zbiór zmiennych globalnych w Warsaw BASICU'u i w interpreterze rozbudowanym dla Czytelników Bajtka opowiemy w następnym odcinku.

Krzysztof Gajewski
Bogusław Radziszewski

PROGRAM 1

```
., C89D A9 A8       LDA #$A8
., C89F A0 C8       LDY #$C8
., C8A1 8D 0A 03    STA $030A
., C8A4 8C 0B 03    STY $030B
., C8A7 60          RTS

., C8A8 A9 00       LDA #$00
., C8AA 85 0D       STA $0D
., C8AC 20 73 00    JSR $0073
., C8AF C9 5C       CMP #$5C
., C8B1 F0 0B       BEQ $C8BE
., C8B3 A5 7A       LDA $7A
., C8B5 D0 02       BNE $C8B9
., C8B7 C6 7B       DEC $7B
., C8B9 C6 7A       DEC $7A
., C8BB 4C 86 AE    JMP $AE86
., C8BE 20 73 00    JSR $0073
., C8C1 20 73 00    JSR $0073
., C8C4 A5 49       LDA $49
., C8C6 48          PHA
., C8C7 A5 4A       LDA $4A
., C8C9 48          PHA
., C8CA 20 07 C7    JSR $C707
., C8CD 68          PLA
., C8CE 85 4A       STA $4A
., C8D0 68          PLA
., C8D1 85 49       STA $49
., C8D3 A9 00       LDA #$00
., C8D5 85 0D       STA $0D
., C8D7 60          RTS

., 030A A8 C8
```

PROGRAM 2

```
800 PRINT "Część 8"
802 X-51357: N-58: C-0
804 FOR I-0 TO N: READ A: POKE X+I.A: C-C+A: NEXT
806 IF C<>5814 THEN PRINT "Błąd w części 8": END
808 SYS X: PRINT "Część 8 OK"
810 DATA 169,168,160.200,141,10,3.140,11,3,96,169
812 DATA 0,133.13,32,115,0,201.92,240.11,165,122
814 DATA 208,2,198,123,198,122,76.134,174,32,115,0
816 DATA 32,115,0,165,73,72.165,74.72,32,7,199
818 DATA 104.133.74.104.133,73,169,0,133,13.96
```

Od red. Warsaw Basic dziś także na kol. 21

MINI-PAKIET

GRAFICZNY

NA KOMPUTER AMSTRAD PCW 8256/8512

(część trzecia — okna)

Jednym z elementów nowoczesnego programowania jest szeroko stosowana technika okien, której podstawą jest możliwość nakładania w dowolnym miejscu na ekran nowych okien i późniejsze ich usuwanie z odtworzeniem poprzedniej zawartości ekranu.

Kompilator Turbo Pascala dla PCW sam z siebie nie daje takich możliwości, ale jeśli napiszemy kilka procedur w tym języku i dodamy zbiór RSX, to otrzymamy zestaw, którym będzie można się posłużyć we własnych programach. Prezentowana w tym odcinku część pakietu graficznego jest zgodna z poprzednimi częściami publikowanymi wcześniej i obejmuje 21 uzytecznych procedur obsługujących ekran JOYCE'a (listing 1. Zbiór WINDOW.SYS). Listing 2 zawiera zbiór WINDOW.MAC, z którego generujemy RSX'a, używanego przez procedury w Pascal-u. Na listingu 3 (zbiór DEMOWIND.PAS) przedstawiono program demonstracyjny — TestWindow, pokazujący sposób posługiwania się pakietem. Na rysunku 1 znajduje się kopia ekranu sporządzona podczas pracy programu TestWindow, a rysunek 2 zawiera, utworzony przez program, wydruk jednego z okien.

Zbiór WINDOW.SYS

Podstawowymi procedurami pakietu, pozwalającymi tworzyć na ekranie JOYCE'a dynamiczne okna są dwie procedury GETLINE i PUTLINE. Pierwsza z nich pobiera do bufora część wiersza (ROW) ekranu ograniczonego z lewej i prawej strony (kolumny C1 i C2). Wiersz składa się z 8 cienkich linii (na całym ekranie mieści się ich 256). Druga z procedur wykonuje czynność odwrotną umieszczając zawartość bufora w odpowiednim miejscu ekranu. Obie procedury wykorzystują RSX'a i bufor przez niego stworzony. Bufor ten musi znajdować się we wspólnej dla wszystkich banków pamięci. Z tego powodu wszystkie rozwiązania nie oparte o RSX'y są kłopotliwe i mało naturalne. Procedura InitGPL, inicjalizująca pakiet poprzez udostępnienie adresu bufora, musi być wywołana na początku programu. Dwie następne procedury W_Save i W_Load, służą do zapamiętania fragmentu ekranu (okna) i do jego późniejszego odtworzenia. W komputerach, które mają duzo pamięci operacyjnej, okna przechowuje się na stosie. W wypadku Amstrada, gdzie obszar pamięci przeznaczony na programy użytkownika (tzw. TPA — *Transient Program Area*) jest mały, musimy skorzystać z Ramdysku. Okno jest zapisywane na zbiór dyskowy, którego nazwa została określona przez stałą TFN.

Dzięki procedurom SP, PrintWindow i HardCopy mozliwe jest drukowanie okien i całego ekranu na drukarce. Procedura SP została napisana w assemblerze i zamienia pewną liczbę bajtów (zmienna q) z postaci ekranowej (zmienna s) na postać drukarkową (zmienna p). Zamiana obejmuje porcje po 8 bajtów. Okna pamiętane są w formie kolejnych bajtów, a nie w formie kodów ASCII kolejnych znaków wyświetlanych na ekranie. Pozwala to zapamiętywać także grafikę, ale z pewnym ograniczeniem. Ekran zawiera 720 punktów na 256, które w trybie tekstowym obejmują 32 wiersze po 90 kolumn. Okno nie może zaczynać się w środku wyświetlanego znaku. Kwantuje to w pewien sposób obszary możliwe do zapamiętania na dysku. Sytuację tę można zmienić, modyfikując procedury assemblerowe RSX'a. Wymaga to jednak dluzszego i mniej przejrzystego kodu.

Do ułatwienia programowania przez definiowanie okien lub znajdowanie ich parametrów służą dwie kolejne procedury: Window i WindowParams. W przypadku okien standardowe procedury Turbo Pascala InsLine i DelLine działają błędnie. Ich zmodyfikowane wersje InsLn i DelLn wykorzystując procedury GetLine i PutLine, działają o jakieś 50% dłużej, ale poprawnie.

Kilka bardzo użytecznych, ale prostych i krótkich procedur i funkcji może w większości pracować także na CPC 6128. Omówimy niektóre z nich, informację o reszcie pozostawiając w komentarzach wewnątrz programu. Procedura Cursor gasi lub zapala kursor na ekranie. Likwidacja lub uaktywnienie linii statusu ("Drive is ...") odbywa się przez wywołanie z odpowiednim parametrem (off lub on) procedury Status. Do wyzerowania ekranu służy procedura RstScr. Czyści ona cały ekran, likwiduje okno, przywraca standardowe barwy tła i atramentu.

Umiejętne wykorzystanie proponowanego zestawu procedur pozwala na pisanie bardzo efektownych i skutecznych programów mogących zadowolić naprawdę wybrednego uzytkownika. Na rys. 3 znajduje się jeden z ekranów profesjonalnego programu, działającego na JOYCE'e z wykorzystaniem omawianego pakietu.

Zbiór Window.MAC

Zbiór ten służy do wygenerowania RSX'a umożliwiającego procedurom pascalowym dostęp do ekranu. Wywołanie RSX'a jako funkcji BDOS z numerem 81 (zawartość rejestru C procesora) pozwala przesłać do 720 bajtów z pamięci ekranu (BANK 0) do bufora znajdującego się w TPA (BANK 1). Transfer odwrotny, tzn. do pamięci ekranu, wykonywany jest dla numeru 82. Wywołanie z wartością C równą 83 zwraca adres bufora. Dzięki automodyfikacji kodu mozliwe było istotne zmniejszenie pamięci wymaganej dla RSX'a.

Przygotowanie samego RSX'a polega na podziałaniu na zbiór WINDOW.MAC zbiorem wsadowym MAKERSX.SUB. Odpowiada to wykonaniu następującej sekwencji poleceń:

A>M80 = window
A>LINK window[op]
A>REN window.rsx=window.prl

Zbiór DEMOWIND.PAS

Przykładowy program demonstrujący mozliwości „okienne" pakietu, zawarty w w zbiorze DEMOWIND.PAS, należy skompilować kompilatorem Turbo Pascala i dołączyć zbiór RSX. Kompilacja na dysk musi odbywać się ze zmniejszonym adresem końcowym (np. C000 hex). Dołączenie RSX'a odbywa się przy pomocy programu GENCOM.COM:

A>GENCOM demownd.com window.rsx

W podobny sposób należy traktować własne programy wykorzystujące pakiet, nie zapominając o dołączeniu zbioru WINDOW.SYS dyrektywą INCLUDE:

(*I WINDOW.SYS *)

Organizacja ekranu komputera Amstrad PCW 8256

Cała pamięć JOYCE'a podzielona jest na 4 banki (ang. BANK) po 64 kB. System operacyjny i pamięć ekranu znajdują się w banku (0) natomiast TPA, część BDOS'a, BIOS'a i strona zerowa rezydu-

ją w banku (1). W tym banku umieszczane są programy użytkownika, a dostęp do pamięci ekranu odbywa się w assemblerze przy pomocy następującej sekwencji rozkazów:
LD BC,code
CALL 0FC5AH
dw 00E9H
gdzie *code* jest adresem procedury obsługującej ekran. Procedura ta musi znajdować się w ostatnim bloku wspólnym dla wszystkich banków (adres: C000-FFFF). Znalezienie adresu bajtu pamięci ekranu (*scradr*) dla punktu o współrzędnych x, y wymaga zastosowania następujących wzorów:
wiersz = y div 8
rolladr = B600 + wiersz * 16
kolumna = x div 8
scradr = 2 * $(rolladr)_M$ + kolumna * 8 + y mod 8
gdzie ( )$_M$ oznacza zawartość pamięci wskazywanej zawartością nawiasu, a div i mod są odpowiednio operatorami dzielenia i reszty dla liczb całkowitych. Odpowiedni bit w bajcie wyznaczany jest wg wzoru:
bit = x mod 8
Podane wzory ułatwią analizę procedur assemblerowych występujących w omawianych RSX'ach.

Mam nadzieję, że prezentowany pakiet uświadomi wszystkim użytkownikom JOYCE'a, jak wartościowym sprzętem dysponują, a osobom zajmującym się pisaniem programów profesjonalnych pozwoli na stworzenie efektownego i pożytecznego oprogramowania na ten 8-bitowy komputer.

Jarosław Młodzki

LISTING 1

```
Listing zbioru A:WINDOW.SYS

(*****************************************************************)
(*                                                               *)
(*                      Zbiór WINDOW.SYS                         *)
(*                          ver. 1.0i                            *)
(*                                                               *)
(*              (C) Jarosław Mlodzki  Czerwiec 1988              *)
(*                                                               *)
(*   Umożliwia realizację podstawowych funkcji obslugi ekranu    *)
(*   lacznie z wykorzystaniem okien tekstowych i graficznych     *)
(*   na komputerach AMSTRAD PCW 8256/8512 w Turbo-Pascalu        *)
(*   pod kontrolą systemu operacyjnego  CP/M Plus.               *)
(*                                                               *)
(* Lista procedur:                                               *)
(*                                                               *)
(*  1. procedure Error (e : byte);                               *)
(*  2. procedure Cursor (s : boolean);                           *)
(*  3. procedure Dark (s : boolean);                             *)
(*  4. procedure Wrap (s : boolean);                             *)
(*  5. procedure Under (s : boolean);                            *)
(*  6. procedure Inverse (s : boolean);                          *)
(*  7. procedure Status (s : boolean);                           *)
(*  8. procedure InitGPL;                                        *)
(*  9. procedure GetLine (row,c1,c2 : integer);                  *)
(* 10. procedure PutLine (row,c1,c2 : integer);                  *)
(* 11. procedure W_Save (x1,y1, x2,y2 : integer);                *)
(* 12. procedure W_Load (x1,y1, x2,y2 : integer);                *)
(* 13. procedure sp (var p,s; q : byte);                         *)
(* 14. procedure PrintWindow (x1,y1, x2,y2 : integer);           *)
(* 15. procedure HardCopy;                                       *)
(* 16. procedure Window (x1,y1, x2,y2 : integer);                *)
(* 17. procedure WindowParams (var x1,y1, x2,y2, xc,yc : byte);  *)
(* 18. Function  WhereX : byte;                                  *)
(* 19. Function  WhereY : byte;                                  *)
(* 20. procedure InsLn;                                          *)
(* 21. procedure DelLn;                                          *)
(* 22. procedure RstScr;                                         *)
(*                                                               *)
(* Uwagi.                                                        *)
(*                                                               *)
(* 1. Wymagany zbiór WINDOW.RSX realizujący odwolania do syste-  *)
(*    mu operacyjnego, które pozwalają na dostep do pamięci      *)
(*    ekranu                                                     *)
(*                                                               *)
(*****************************************************************)
const
     TFN = 'm:TMP.$$W';    (* tymczasowa nazwa zbioru *)
     off = false;
     on  = true;
type
  buffer = record
             sline : array (.0..719.) of byte;
             prms  : array (.1..24.)  of integer;
           end;
var
     pbuf : ^buffer;
     wf   : file;

procedure Error (e : byte);
(****************************************)
(* Obsluga bledów                       *)
(****************************************)
begin
  writeln (£7'Blad ',e,
          ' w programie. Przerwano.');
  halt;
end; (* Error *)

procedure Dark (s : boolean);
(****************************************)
(* Zmiana tła i atramentu.              *)
(****************************************)
begin
     case s
     of false : write(£27'c'£0£27'b'£63);
        true  : write(£27'b'£0£27'c'£63);
     end;
end; (* of Dark *)

procedure Cursor (s : boolean);
(****************************************)
(* Włączanie i wyłączanie kursora       *)
(****************************************)


begin
    case s
    of false : write(£27'f');
       true  : write(£27'e');
    end;
end; (* Cursor *)

procedure Wrap (s : boolean);
(****************************************)
(* Włączanie i wyłączanie trybu obsługi *)
(* ostatniego znaku na ekranie. Jeśli   *)
(* włączony to ostatni znak w oknie druko-*)
(* wany będzie w tym samym miejscu, nie *)
(* powodując przesunecia ekranu o jedną *)
(* linię (tzw. scroll)                  *)
(****************************************)
begin
    case s
    of false : write(£27'w');
       true  : write(£27'v');
    end;
end; (* Wrap *)

procedure Under (s : boolean);
(****************************************)
(* Włączanie i wyłączanie podkreśalenia *)
(****************************************)
begin
    case s
    of false : write(£27'u');
       true  : write(£27'r');
    end;
end; (* Under *)

procedure Inverse (s : boolean);
(****************************************)
(* Wlaczanie i wyłączanie podświetlenia *)
(****************************************)
begin
    case s
    of false : write(£27'q');
       true  : write(£27'p');
    end;
end; (* Inverse *)

procedure Status (s : boolean);
(****************************************)
(* Włączanie i wyłączanie linii statusu: *)
(* Drive is A:                          *)
(****************************************)
begin
    case s
    of false : write(£27'0');
       true  : write(£27'1');
    end;
end; (* Status *)

procedure InitGPL;
(******************************************)
(* Inicjalizacja pakietu. Musi być wywołana *)
(* przed owołaniem do procedur realizującychh*)
(* okna                                   *)
(******************************************)
begin
     pbuf := PTR (BdosHL (83) );
end; (* of InitGPL  *)

procedure GetLine ( ROW, c1,c2 : integer);
(******************************************)
(* Pobiera do bufora w TPA jedną linię ROW *)
(* ekranu (8 cienkich linijek) od kolumny  *)
(* c1 do c2                               *)
(******************************************)
begin
     with pbuf^
     do begin
             prms (.1.) := ROW - 1;
             prms (.2.) := (c1 - 1) shl 3;
             prms (.3.) := (c2 - c1 + 1) shl 3;
             Bdos (81);
        end;
end; (* GetLine *)
```

```
procedure PutLine (row,cl,c2 : integer);
(******************************************)
(* Przesyła z bufora w TPA do pamięci ekranu *)
(* jedną linię ROW między kolumnami cl i c2  *)
(******************************************)
begin
   with pbuf^
   do begin
         prms (.1.) := row - 1;
         prms (.2.) := (cl - 1) shl 3;
         prms (.3.) := (c2 - cl + 1) shl 3;
         Bdos (82);
      end;
end; (* PutLine *)

procedure W_Save (xl,yl, x2,y2 : integer);
(******************************************)
(* Zapisuje na dysk (M:) okno o wsp.:    *)
(* lewy  górny róg xl,yl,                *)
(* prawy dolny róg x2,y2.                *)
(******************************************)
var row : integer;
begin
   assign (wf,TFN);
   rewrite(wf);
   for row := yl to y2
   do begin
         GetLine    (row, xl,x2);
         BlockWrite (wf,pbuf^,6);
      end;
   close(wf);
end; (* W_save *)

procedure W_Load (xl,yl, x2,y2 : integer);
(******************************************)
(* Laduje okno z dysku na ekran          *)
(******************************************)
var row : integer;
begin
   assign (wf,TFN);    (*$I-*)
   reset  (wf);        (*$I+*)
   if IOresult<>0
   then Error(1);
   for row := yl to y2
   do begin
         BlockRead (wf,pbuf^,6);
         PutLine   (row, xl,x2);
      end;
   close(wf);
end; (* W_Load *)

procedure sp (var p,s; q : byte);
(******************************************)
(* Zamienia linię ekranu, na linie pozwa- *)
(* lajaca na wydruk w trybie graficznym   *)
(* na drukarce                            *)
(******************************************)
begin
  inline ( $ED/$5B/s/  $2A/p/  $3A/q/
      $47/$C5/$06/$08/$C5/$E5/$1A/$05/$08/$CB/
      $07/$CB/$16/$23/$10/$F9/$E1/$13/$C1/$10/
      $EF/$01/$08/$00/$09/$C1/$10/$E5 );
end; (* screen line to printer line *)

procedure PrintWindow (xl,yl, x2,y2 : integer);
(******************************************)
(* Drukuje Okno na drukarce               *)
(******************************************)
var
   i, row, q8 : integer;
   buf : array (.1..720.) of byte;
   q : byte;
begin
  write (lst,£27£51£24);
  for row := yl to y2
  do begin
       GetLine (row, xl,x2);
       q := x2 - xl + 1;
       q8 := 8 * q;
       SP (buf, pbuf^.sline, q);
       write (lst, £27'L',chr(Lo(q8)),
                          chr(Hi(q8)));
       for i := 1 to q8
       do write (lst,chr(buf(.i.)));
       writeln  (lst);
     end;
  write(lst,£27£48);
end; (* Print Window *)

procedure HardCopy;
(******************************************)
(* Zrzuca cały ekran na drukarkę          *)
(******************************************)
begin
   PrintWindow (1,1,90,32);
end; (* HardCopy *)

procedure Window (xl,yl, x2,y2 : integer);
(******************************************)
(* otwiera na ekranie okno                *)
(******************************************)
var
  tr, clm, h, w : byte;
begin
   tr  := yl + 31;      h   := y2 - yl + 32;
   clm := xl + 31;      w   := x2 - xl + 32;
   write (£27'X',chr(tr),chr(clm),
                 chr(h),chr(w));
end; (* Window *)

procedure WindowParams (var xl,yl, x2,y2,
                              xc,yc : byte);
(******************************************)
(* Wyznacza parametry aktualnego okna:   *)
(* 1. xl,yl - wsp. lewego górnego rogu   *)
(* 2. x2,y2 - wsp. prawego dolnego rogu  *)
(* 3. xc,yc - wsp. kursora wzg. okna     *)
(******************************************)
var
   Uxy,Dxy,Cxy : integer;
begin
  InLine ($CD/$5A/$FC/ $BF/$00/$ED/$43/Uxy/
          $ED/$53/Dxy/ $22/Cxy);
  xl := Lo (Uxy) + 1;   yl := Hi (Uxy) + 1;
  x2 := Lo (Dxy) + xl;  y2 := Hi (Dxy) + yl;
  xc := Lo (Cxy) + 1;   yc := Hi (Cxy) + 1;
end; (* Window PArams *)

Function WhereX : byte;
(******************************************)
(* podaje wsp. X kursora wzg. okna        *)
(******************************************)
var xy : integer;
begin
   InLine ($CD/$5A/$FC/$BF/$00/$22/xy);
   WhereX := Lo (xy) + 1;;
end; (* Where X *)

Function WhereY : byte;
(******************************************)
(* podaje wsp. Y kursora wzg. okna        *)
(******************************************)
var xy : integer;
begin
   InLine ($CD/$5A/$FC/$BF/$00/$22/xy);
   WhereY := Hi (xy) + 1;
end; (* Where X *)

procedure InsLn;
(******************************************)
(* Poprawiona wersja pascalowej procedury *)
(* INSLINE źle działającej na PCW 8256    *)
(******************************************)
var
   xl,yl, x2,y2, xc,yc, i : byte;
begin
   WindowParams (xl,yl, x2,y2, xc,yc);
   if (xl=1) and (x2=90)
   then InsLine
   else begin
          i := y2;
          while i >= yc+yl - 1
          do begin


                GetLine (i-1, xl,x2);
                PutLine (i,   xl,x2);
                i := i - 1;
             end;
          write(£13); clreol;
          GotoXY (xc,yc);
        end;
end; (* InsLn *)

procedure DelLn;
(******************************************)
(* Poprawiona wersja pascalowej procedury *)
(* DELLINE źle działającej na PCW 8256    *)
(******************************************)
var
   xl,yl, x2,y2, xc,yc, i : byte;
begin
   WindowParams (xl,yl, x2,y2, xc,yc);
   if (xl=1) and (x2=90)
   then DelLine
   else begin
          write (£13);    ClrEol;
          i := yc+yl - 1;
          while i<y2
          do begin
                GetLine (i+1, xl,x2);
                PutLine (i,   xl,x2);
                i := i + 1;
             end;
          GotoXY (1,y2);  ClrEol;
          GotoXY (xc,yc);
        end;
end; (* DelLn *)

procedure RstScr;
(******************************************)
(* Reset ekranu, zerowanie do sytuacji po-*)
(* czątkowej po włączeniu. Nie odtwarza   *)
(* linii statusu                          *)
(******************************************)
begin
   InLine ($CD/$5A/$FC/$C2/$00);
end; (* Reset Screen *)

(* ------------------------------------------- *)
```

LISTING 2

Listing zbioru A:WINDOW.MAC

```
; ******************************
; *                            *
; *     Zbiór WINDOW.MAC       *
; *        ver. 1.0            *
; *                            *
; *   (C) J. Młodzki 1988      *
; *                            *
; * RSX do pakietu WINDOW.SYS  *
; *                            *
; *  Służy do utworzenia zbioru*
; *       WINDOW.RSX           *
; *                            *
; ******************************

screen  equ  00E9H
xbios   equ  0FC5AH
roller  equ  0B600H
        .z80
        cseg
; rsx header
        ds      6
        jp      start
bdos:   jp      0
        dw      0
        db      0FFH,0,'WINDOWSY'
        db      0,0,0
start:  ld      a,c
        cp      81
        jp      z,GET
        cp      82
        jp      z,PUT
        cp      83
        jp      z,ADRES
        jp      bdos
adres:  ld      hl,buffer
        ret
GET:    ld      a,0
        jr      cont
PUT:    ld      a,0EBH ; ex de,hl
cont:   ld      (opex),a
        ld      hl,(row)
        ld      de,roller
        add     hl,hl
        add     hl,hl
        add     hl,hl
        add     hl,hl
        add     hl,de
        ld      bc,routin
        call    xbios
        dw      screen
        ret
routin: ld      a,(hl)
        inc     hl
        ld      h,(hl)
        ld      l,a
        add     hl,hl
        ld      de,(cs)
        add     hl,de
        ld      bc,(qse)
        ld      de,buffer
opex:   db      0
        ldir
        ret
buffer: ds      720
row:    dw      0
cs:     dw      0
qse:    dw      0
        ds      50
        end

; ******************************
```

LISTING 3

Listing zbioru DEMOWIND.PAS

```
program TestWindow;   (*$C-,U-*)
(**************************************************)
(*                                                *)
(*    Turbo Pascal Window Demo na PCW 8256/8512   *)
(*                                                *)
(*           (C)  JM  Czerwiec 1988               *)
(*                                                *)
(**************************************************)
const
    Windows    = 3;
    Vtab       : array(.1..Windows,1..5.) of Integer
                 = (( 5,  3,  35, 11,  1),
                    (45,  3,  75, 11,  1),
                    ( 5, 15,  75, 23,  1));
                (* X0, Y0,  X1, Y1, LineNo *)
type
    String255  = String(.255.);
var
    i          : Integer;    Ch          : Char;

(* $I WINDOW.SYS *)

procedure Frame ( UpperLeftX, UpperLeftY,
                  LowerRightX, LowerRightY: Integer);
(**************************************************)
(* Rysuje ramke wokól wybranego okna              *)
(**************************************************)
var
      i: Integer;
begin
      GotoXY (UpperLeftX, UpperLeftY);              Write(£150);
      for i:= UpperLeftX+1 to LowerRightX-1 do Write(£154);
                                                    Write(£156);
      for i:= UpperLeftY+1 to LowerRightY-1 do
      begin
         GotoXY (UpperLeftX , i);  Write(£149);
         GotoXY (LowerRightX, i);  Write(£149);
      end;
      GotoXY (UpperLeftX, LowerRightY);             Write(£147);
      for i:= UpperLeftX+1 to LowerRightX-1 do Write(£154);
                                                    Write(£153);
end  (* Frame *);

function RanStr(Len: Integer): String255;
(**************************************************)
(* Produkuje przypadkowy ciag znakow o dlugosci Len *)
(**************************************************)
var
      S: String255;       i: Integer;
begin
      S(.0.):=Chr(Len);
      for Len:=1 to Len do
         S(.Len.):=Chr(Random(223)+32);
      RanStr:=S;
end  (* RanStr *);

procedure SelectWindow(Win: Integer);
(**************************************************)
(* Wybiera jedno z zadeklarowanych okien          *)
(**************************************************)
begin
      Window ( Vtab(.Win,1.), Vtab(.Win,2.),
               Vtab(.Win,3.), Vtab(.Win,4.) );
end  (* SelectWindow *);

procedure InsInWindow (w : byte);
(**************************************************)
(* Wybiera okno i wstawia w nie kolejne linie     *)
(**************************************************)
var
   L : integer;
begin
      if w=3 then L:= 58 else L:=18;
      SelectWindow (w);
      GotoXY (2,1);
      LowVideo;
      Write('Line ', Vtab(.w,5.):5, ' ', RanStr(L));
      InsLn;
      Vtab(.w,5.):=Succ(Vtab(.w,5.));
      NormVideo;
end  (* InsInWindow *);


procedure DelInWindow (w : byte);
(**************************************************)
(* Wybiera okno i wyrzuca z niego kolejne linie   *)
(**************************************************)
var
   L : integer;
begin
      if w=3 then L:= 58 else L:=18;
      SelectWindow (w);
      GotoXY (2,1);
      LowVideo;
      Write('Line ', Vtab(.w,5.):5, ' ', RanStr(L));
      DelLn;
      Vtab(.w,5.):=Succ(Vtab(.w,5.));
      NormVideo;
end  (* DelInWindow *);

procedure FillWindow (w : byte);
(**************************************************)
(* Wybiera okno, i wypelnia je przypadkowym znakiem *)
(**************************************************)
var
   L,i  : byte;
   ch   : char;
   x1,y1, x2,y2, xc,yc : byte;
begin
      if w=3 then L := 71 else L := 31;
      ch := chr(random(222)+32);
      SelectWindow (w);
      GotoXY (1,1);
      LowVideo;
      repeat
            write(ch);
            WindowParams (x1,y1,x2,y2,xc,yc);
      until  (y2=yc+y1-1);
      wrap(off);
      for i := 1 to L  do write(ch);
      wrap(on);
      NormVideo;
end  (* FillWindow *);

procedure SwitchWindow (i : byte);
(**************************************************)
(* Wybiera okno, chowa je na ramdysk, czysci, a   *)
(* nastepnie ponownie odtwarza                    *)
(**************************************************)
begin
      SelectWindow (i);
      W_Save (Vtab(.i,1.)-1, Vtab(.i,2.)-1,
              Vtab(.i,3.)+1, Vtab(.i,4.)+1);
      clrscr;
      W_Load (Vtab(.i,1.)-1, Vtab(.i,2.)-1,
              Vtab(.i,3.)+1, Vtab(.i,4.)+1);
end  (* SwitchWindow *);

begin           (* MAIN *)
   Init6PL;
   clrscr;       Status(off);  Cursor(off);  Wrap(on);

   GotoXY (1,1);
   Write('123456789 123456789 123456789 123456789 ',
         '123456789 123456789 123456789 123456789 ');

   for i:=1 to 30
   do begin
         GotoXY (1,i);   write(i);
      end;

   Under(on);
   GotoXY (10,27);
   Write('TURBO PASCAL Window Demo (PCW 8256) ',
         '-  Nacisnij dowolny klawisz');
   Under(off);

   for i:=1 to Windows
    do Frame (Vtab(.i,1.)-1, Vtab(.i,2.)-1,
              Vtab(.i,3.)+1, Vtab(.i,4.)+1);

   repeat
      InsInWindow(1);
      InsInWindow(2);
      InsInWindow(3);

   until KeyPressed;        Read(KBD, Ch);

   repeat
      DelInWindow(1);
      DelInWindow(2);
      DelInWindow(3);
   until KeyPressed;        Read(KBD, Ch);

   repeat
      FillWindow(1);
      FillWindow(2);
      FillWindow(3);
   until KeyPressed;        Read(KBD, Ch);

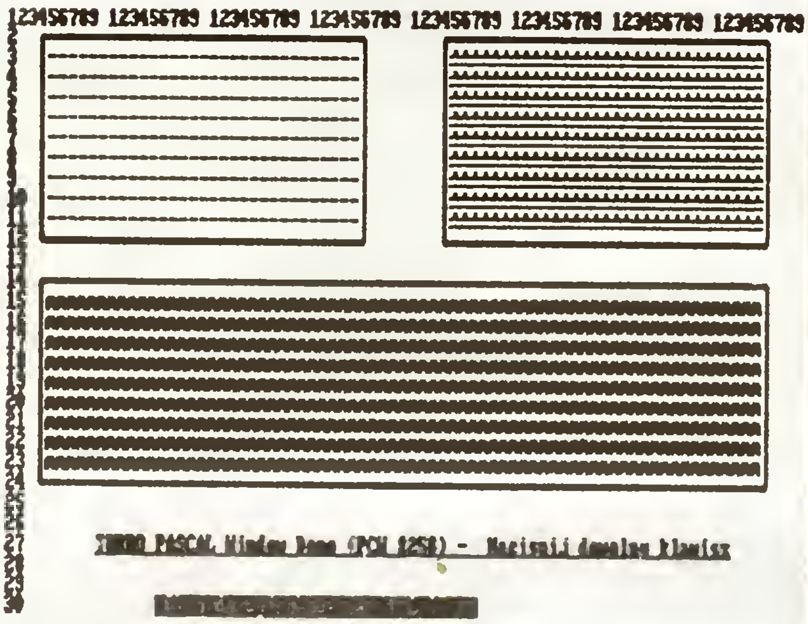
   repeat
      SwitchWindow(1);
      SwitchWindow(2);
      SwitchWindow(3);
   until KeyPressed;        Read(KBD, Ch);
```

```
window(1,1,90,31);     Cursor(on);    Inverse(on);

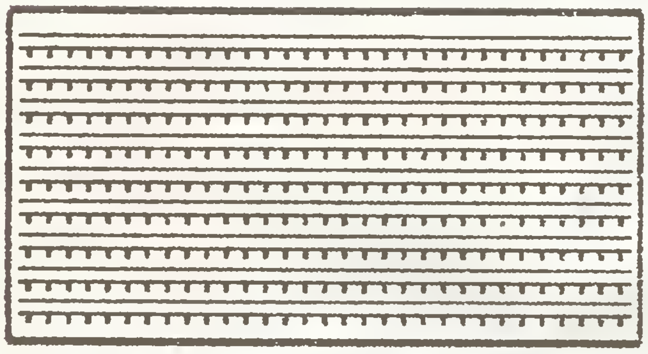
GotoXY (16,30);
write (' Okno do wydrukowania (1,2,3): ');
readln(ch);
i := ord(ch)-48;
if (i<1) or (i>3)
then begin
       Inverse(off);  clrscr;   Status(on);
       exit;
     end;

PrintWindow (Wtab(.i,1.)-1, Wtab(.i,2.)-1,
             Wtab(.i,3.)+1, Wtab(.i,4.)+1);

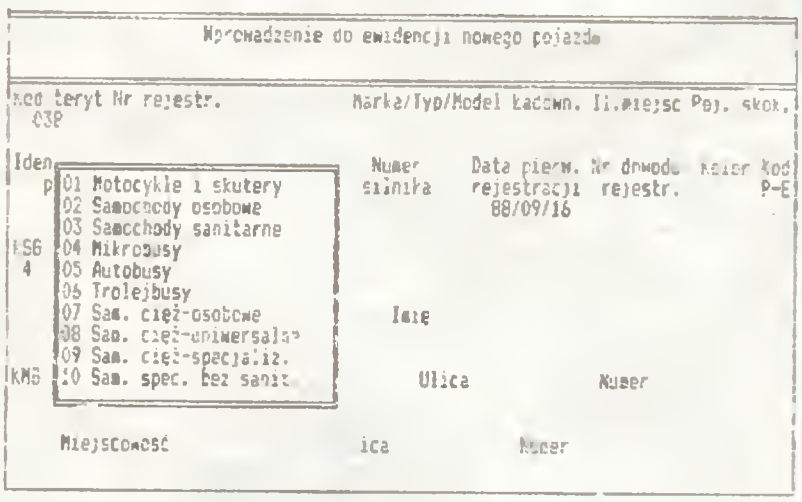
clrscr;             Status(on);    Inverse(off);
end.
```



Rys. 1. Przykładowy wygląd ekranu programu DEMOWIND



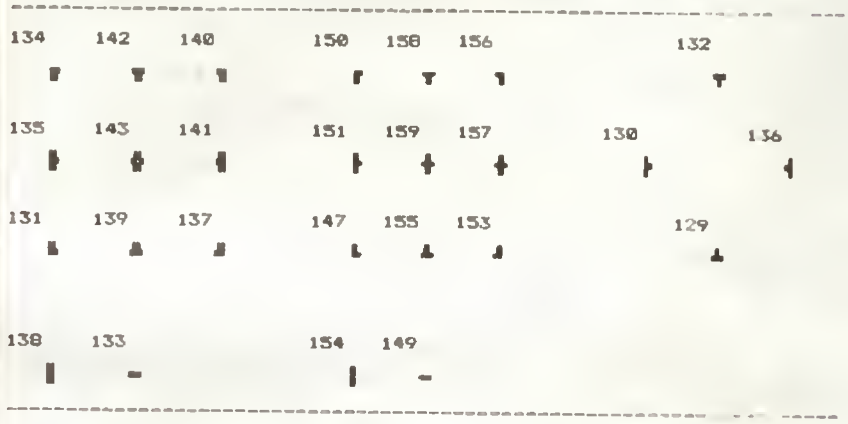
Rys. 2. Kopia drugiego okna na drukarce



Rys. 3. Ekran programu korzystającego z okien

ramki na PCW

Podstawowy podzbiór kodu znakowego ASCII składa się ze 128 elementów i zawiera: znaki sterujące, duże i małe litery, cyfry, znaki interpunkcyjne i niektóre symbole matematyczne. Pozostałe 128 możliwych kombinacji 8-bitowego słowa jest wykorzystywane w mniej standardowy sposób i dosyć dowolnie implementowane przez producentów sprzętu. Najczęściej wśród wielu innych znaków znajdują się znaki semigraficzne umożliwiające tworzenie w trybie tekstowym ramek. Na komputerze AMSTRAD PCW są to kody od 128 do 159. Ponieważ zapamiętanie ich w kolejności jakiej występują może być dosyć trudne, zebrano je w bardziej przejrzystej formie (patrz rys. 1).



Rys. 1. Kody elementów ramek na PCW.

Zestawienie to będzie dużym ułatwieniem przy pisaniu programów typu:

writeln(#134#138#138#138#140);
writeln(#133#032#032#032#133);
writeln(#131#138#138#138#137);   (* PASCAL *).

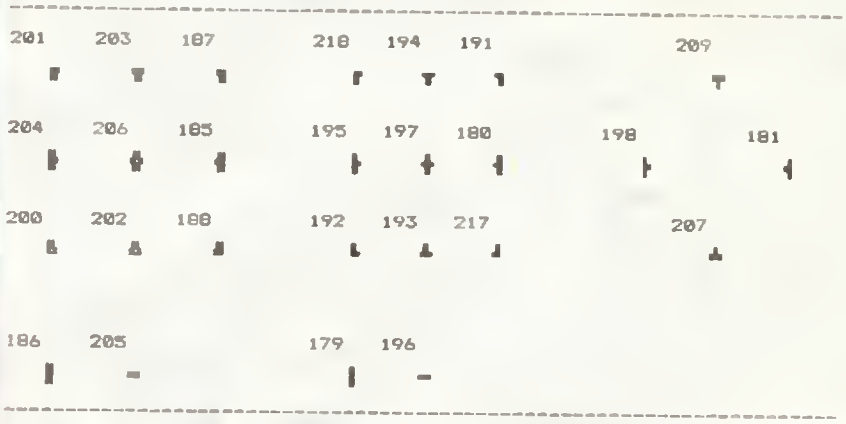
lub

PRINT CHR$(134);CHR$(138);CHR$(138);CHR$(138);CHR$(140)
PRINT CHR$(133);CHR$(032);CHR$(032);CHR$(032);CHR$(133)
PRINT CHR$(131);CHR$(138);CHR$(138);CHR$(138);CHR$(137)
(* BASIC *).

W wyniku działania powyższego ciągu instrukcji otrzymamy następującą ramkę:



Dla osób przenoszących oprogramowanie między AMSTRADem PCW a IBM'em PC równie użyteczne będzie zestawienie w podobnej formie tych samych znaków w komputerze IBM PC (rys. 2). Na tym sprzęcie w odróżnieniu do PCW, znaki te dostępne są z klawiatury. Trzymając wciśnięty klawisz ALT naciskamy kolejne cyfry kodu znaku. Sekwencja 2,0,3 da



Rys. 2. Te same kody na IBM PC

HACKER

1 CHINY



CHIŃSKI MUR

dajesz
dostajesz:
- 8 część mapy
- a ming vase
- a jade carving

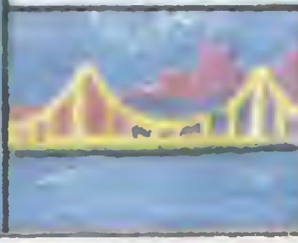
2 JAPONIA



OSAKA

dajesz
dostajesz
- 6 część mapy
- cultured pearls
- a 35 mm camera

3 USA



SAN FRANCISCO

dajesz
dostajesz
- 4 część mapy
- gold nuggets
- 49er season ticket

4 USA



NOWY YORK

dajesz
dostajesz:
- [illegible]
- [illegible] dimonds
- stock & bonds

5 USA



WASZYNGTON

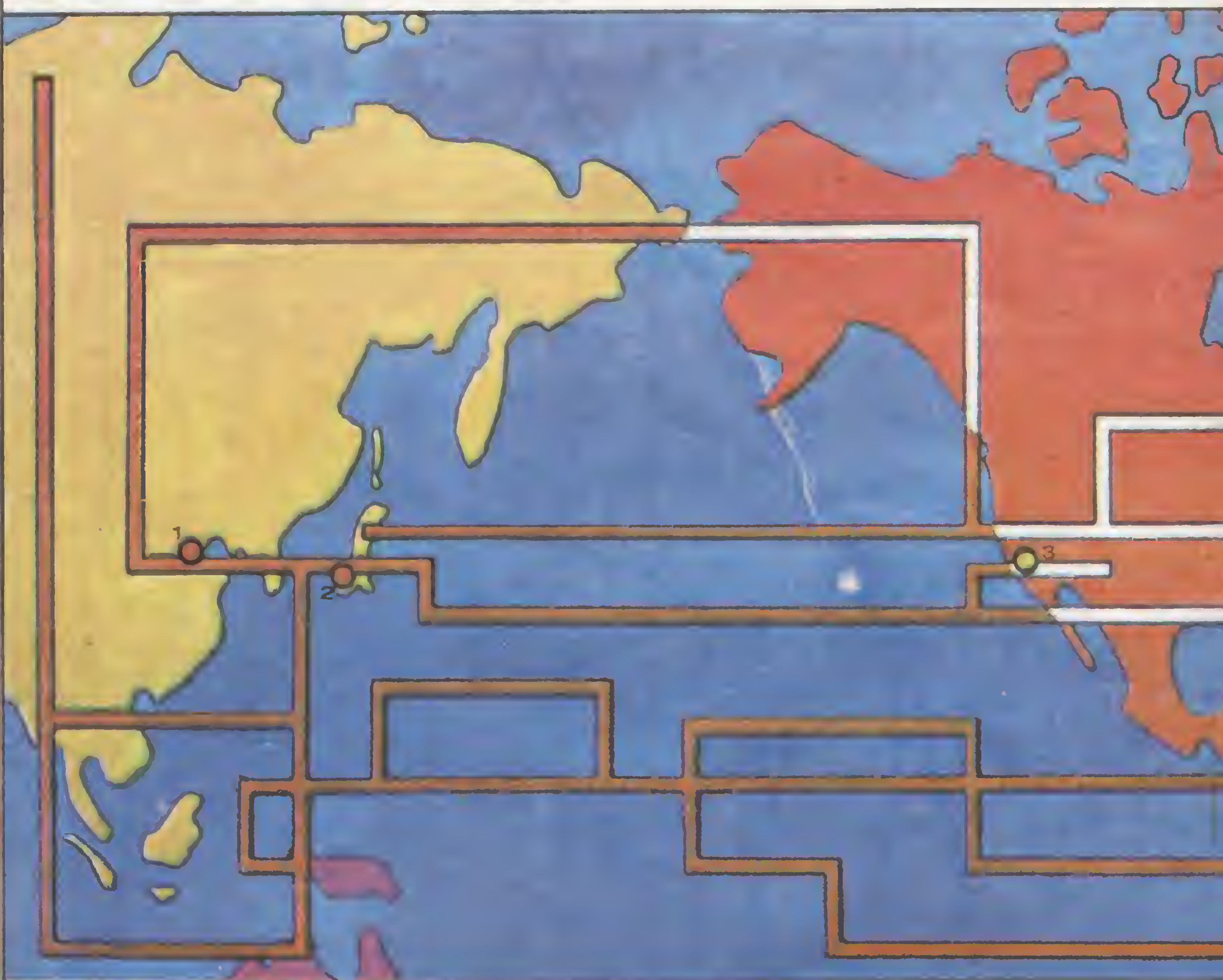
dajesz
dostajesz

W przygotowaniach do odlotu na Proximę α, w celu spędzenia tam zasłużonego urlopu, przeszkodził ci dźwięk videofonu. Na monitorze ukazał się sekretarz generała Roberta Mallory'ego, szefa kontrwywiadu Federacji Planet Sprzymierzonych. Od razu miałeś przeczucie, że stało się coś złego. Potwierdzały to słowa sekretarza, który powiedział, że generał chce bezzwłocznie się z Tobą widzieć. Poszedłeś do pokoju szefa. Okazało się, że Twoje przeczucie nie myliło cię. Jako najlepszy agent zostałeś wybrany do bardzo trudnego zadania, od którego zależy los Federacji.

Z tajnej bazy zostały skradzione dokumenty dotyczące planu MAGMA. Są one bardzo ważne dla Federacji, gdyż jest w nich mowa o zniszczeniu Ziemi, która jest stolicą FPS. Na Ziemi bowiem znajduje się większość laboratoriów i urządzeń, od których zależy normalne funkcjonowanie Federacji. Poleciałeś więc na Księżyc i założyłeś tam tymczasową bazę operacyjną.

W tej trudnej misji, której celem jest odzyskanie planów, ma pomóc ci robot typu SRU. Jest to najnowocześniejsza maszyna tego typu wyprodukowana przez NASA i przeznaczona do zadań specjalnych. Przed wysłaniem SRU na Ziemię, powinieneś go przetestować. W tym celu musisz naprowadzić celownik na odpowiednią część robota i nacisnąć FIRE. Po przetestowaniu robota, wysyłasz go na Ziemię. Będziesz nim sterował zdalnie ze swojej bazy.

Poruszając się tunelami, znajdującymi się pod powierzchnią planety, musisz odnajdywać miasta, wychodzić na powierzchnię i targując się ze szpiegami odkupić od nich wszystkie części planu MAGMA. Części planu jest dziesięć i każdy ze szpiegów ma po jednej. Złodzieje po sprzedaniu swojego kawałka oferują ci różne przedmioty, kupuj je. Mogą przydać się przy transakcji z innymi szpiegami. Pamiętaj, że za jedną część planu możesz dać maksymalnie 5000 $.





Do tego samego co Ty, dążą szpiedzy Unii Niszczycieli. Chcą oni jednak nie uratować Ziemię, lecz ją zniszczyć. Dlatego powinieneś się spieszyć. Musisz uważać też na satelity szpiegowskie, które krążą nad planetą. Gdy dostaniesz się w ich zasięg, komputer zadaja ci pytanie. Jeżeli udzielisz błędnej odpowiedzi, twój SRU zostanie zniszczony.

Czytaj uważnie komunikaty, gdyż mogą okazać się one bardzo ważne. Gdy skompletujesz wszystkie części planu MAGMA i dostarczysz je do Waszyngtonu, ocalisz Ziemię od losu, który chciała zgotować jej Unia Niszczycieli, i tym samym ukończysz grę.

Do twojego wyposażenia dołączono także instrukcje obsługi robota, która zawiera klawisze sterujące SRU oraz opis elementów identyfikacyjnych:

U — wychodzenie na powierzchnię
I — włączanie emitera podczerwieni, który umożliwia widzenie w ciemności
M — umożliwia odbieranie komunikatow "MSG"
C — sygnał przywołujący szpiega
D — zjazd do tunelu
I — zmiana oferty dla szpiega
Y — zgoda na kupno
N — odmowa kupna
E — przejrzenie dotychczasowo zgromadzonych fragmentow planu MAGMA

Elementy identyfikacyjne SRU:

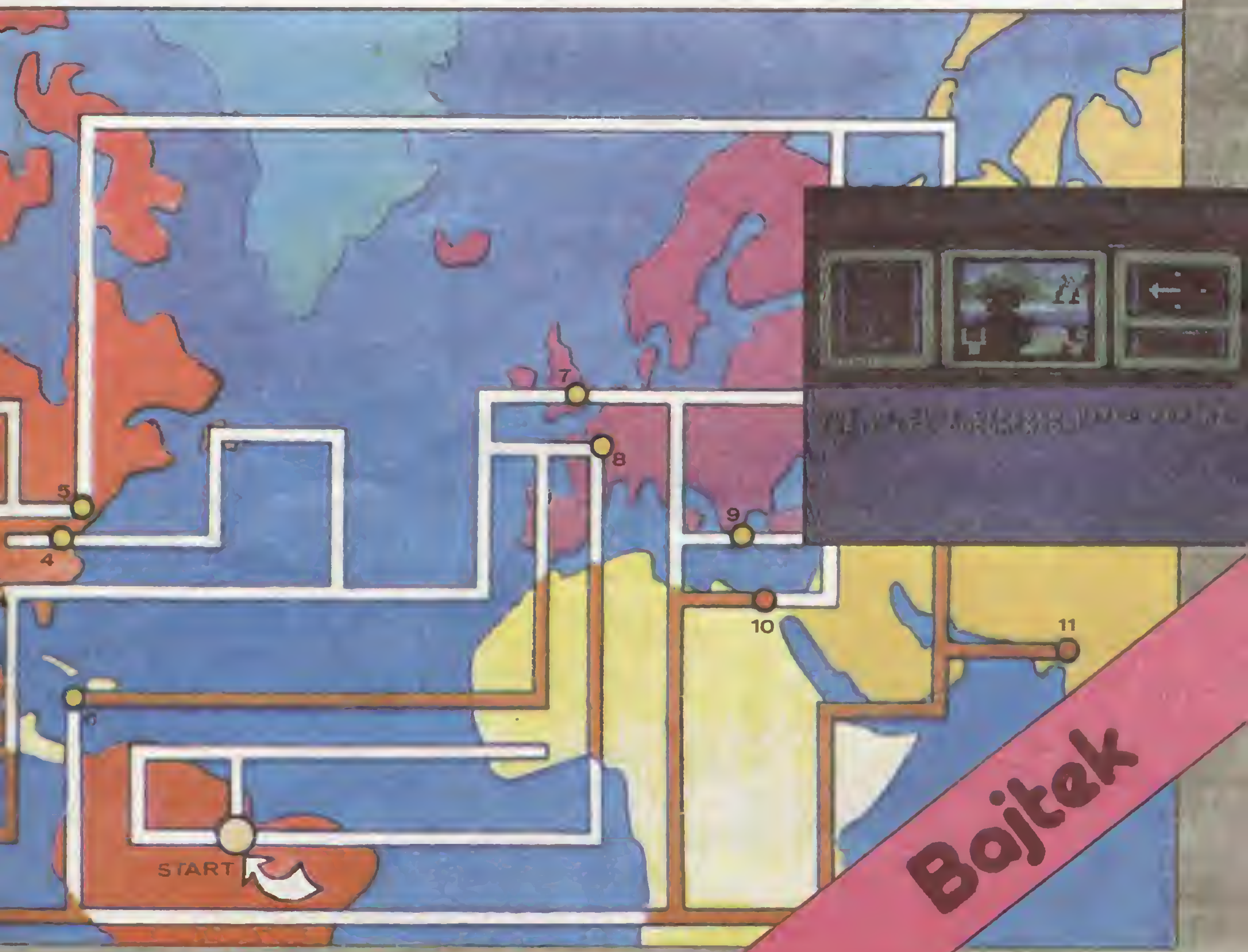
1. INFRARED VIDEO IMAGE SENSOR — ośrodek centralno-wizyjny
2. ASYNCHRONUS DATA COMPACTOR — główny receptor danych dotyckowych
3. HYDRAULIC MOTIVATOR — główny ośrodek napędowy
4. PHLASMON JOINT — plazmowy amartyzator kierunkowy
5. THELMAN PORT — złącze przyłączeniowe urządzeń zewnętrznych

Oto kilka przydatnych wskazówek:

1. Jeżeli na początku gry wpiszesz hasło "AUSTRALIA", to nie będziesz musiał testować robota.
2. Szpiedzy oferują dużo przedmiotów. Nie kupuj wszystkich, gdyż szybko skończą ci się pieniądze.
3. Podczas sprawdzania robota nie możesz popełnić żadnego błędu, bo będziesz musiał testować SRU jeszcze raz.
4. Grę zacznij od Paryża, a następnie udaj się do Londynu.
5. Nigdy nie zatrzymuj się w obszarze stałego zagrożenia, gdyż często kończy się to śmiercią.

Komputer: ZX Spectrum 48/+, Commodore 64/128, Atari XL/XE

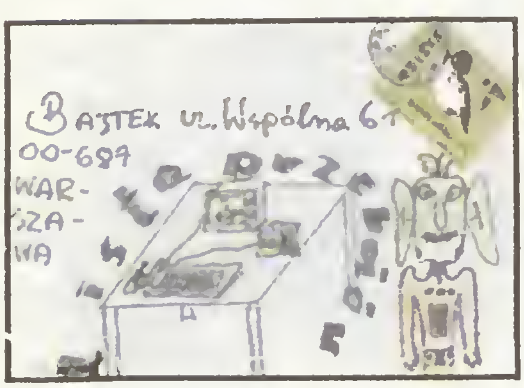
Piotr Kowalczyk
Łukasz Czekajewski



10 BAJTKOWA LISTA PRZEBOJÓW (12/88)

Niespodziewanie zniknął EXPLODING FIST III, za to pojawił się lecz wciąż dobry HACKER, którego „rozgryźli" nasi gracze. Na pierwszym miejscu CHIPWAR, mapa już wkrótce. Adrian Mole traci popularność, podobnie jak Johnny Walker, za to pojawiła się prawie wykopaliskowa PYRAMANIA (hit roku 1983). Czterej komandosi windują się w górę, zobaczymy, jak wysoko wejdą. Tym razem aż 3819 propozycji Złotej Dziesiątki, ulubionych gier jest 241.

			ATARI	AMSTRAD	COMMODORE	SPECTRUM
1	CHIPWAR	↑		×	×	
2	DETECTIVE	↑			×	
3	HACKER	!	×	×	×	×
4	WEST BANK	↑		×	×	×
5	SKYFOX	↓	×	×	×	×
6	STRIKE FORCE COBRA	↑		×	×	×
7	SECRET DIARY	↓		×	×	×
8	NOSFERATU	↑		×	×	×
9	RENEGADE	!		×	×	×
10	FYRAMANIA	!		×		×



Kopertę nadesłała Kamola G z Lublina



THE ROCKY HORROR SHOW

Noc była ciemna i burzliwa. Padał rzęsisty deszcz, niebo co chwilę rozświetlały błyskawice. Droga była mokra i Brad nie mógł jechać nawet 50 mph. Janet spała niespokojnym snem obok niego. Mijali wioski i miasteczka, od dawna uśpione. Nagle wozem zarzuciło i koziołkując wpadł do rowu. Pękły naraz dwie opony. Dalsza podróż była niemożliwa. Trzeba poczekać do rana i pójść po pomoc. Tymczasem dobrze by było gdzieś przenocować.

W oddali stał stary dwór. Na pukanie odpowiedziała tylko sowa. Brad ostrożnie uchylił wielkie i ciężkie podwoje. W środku było ciemno. Oboje ostrożnie weszli do wnętrza. Brad zapalił zapałkę. Gdy nikły płomyk rozświetlił nieco przestronne wnętrze, Brad z przerażeniem spostrzegł, że Janet zniknęła.

Ostrożnie ruszył naprzód. Długo szedł korytarzem. Gdy otworzył ostatnie drzwi, znalazł się w jasnej, przestronnej sali. Pod ścianą stała zbroja, na ścianach wisiały obrazy, z sufitu zwisały żyrandole. Na podłodze błyszczały rozrzucone kawałki miedziorytu. Gdzie niegdzie leżały klucze, w oddali głośno tykał stary zegar. Sala połączona była z drugą i trzecią, po schodach wchodziło się na obszerne piętro. Na ścianie wisiał termometr, pokazujący 20 stopni poniżej zera. Temperatura wciąż rosła. W jednym z pokoi, za zasłoną, Brad postanowił pozbierać rozrzucone kawałki i poukładać pod zasłoną. Może to coś zmieni.

To prawda. Jedynym sposobem na wydostanie się z tego nawiedzonego domu jest skompletowanie rozbitego miedziorytu. W przeszłości dwór był zamieszkany przez dwie rodziny: Eddiego z żoną oraz Teddy'ego z żoną i córką. Eddie był fotografem, a żona Teddy'ego motocyklistką. Na skutek zawiści rodziny te otruły się wzajemnie i teraz po domostwie krążą ich duchy. Eddie z aparatem, z którego strzela, zamknięty jest w lodówce. Gdy się rozmrozi, wyjdzie i będzie próbował zabić Brada. Lodówkę można domrozić uruchamiając przycisk w lodowej komnacie. Temperaturę wskazuje termometr.

Na wykonanie zadania przeznaczone jest 100 jednostek czasu. Każda ma długość ok. 15 sekund. Czasu nie da się zatrzymać w żaden sposób.

Wewnątrz dworu czynne są dwie zapory elektryczne, powodujące natychmiastową śmierć. Można je wyłączyć przyciskiem, umieszczonym nad drabiną, lecz tylko na chwilę. Spotkanie ze zjawą żony Eddiego jest niebezpieczne — zabiera ona ubrania i ukrywa je w jednym z pokoi. Trzeba się więc zasłaniać, a jak przy tym wejść na drabinę ..

Brad musi także uważać na inne zjawy. Nierzadko wyrażają one swoje poglądy co do życia lub innych zjaw. Bardzo przyjemne jest natomiast wciśnięcie przycisku podpisanego „Do Not Press"

Firma: CRL GROUP PLC
Komputer: ZX Spectrum 48/+, Commodore 64/128

(mp)

KRÓL I KRÓLOWA GIER



Krzysztof Grunert uczeń klasy VIIB SP nr 27 w Gdyni
Posiadany komputer: Timex
Ulubiona gra: Match Point
Hobby: modelarstwo, nauka angielskiego



Iwona Stanczyk uczennica klasy VIB SP nr 164 w Warszawie
Posiadany komputer: Spectrum
Hobby: muzyka młodzieżowa, aerobic, pływanie, a od 3 lat także nauka angielskiego



WORLD SERIES BASKETBALL

Zapewne każdy zdaje sobie sprawę, jak trudno jest zrobić grę sportową, w której biorą udział całe drużyny. Niektórzy wiedzą także, jak trudno jest w nią potem grać. Podstawowym problemem jest to, którym graczem mamy sterować. Jeżeli jest ich dwóch w drużynie (jak na przykład w Bump Set Spike), sterowanie przełączane klawiszami 1 i 2. Sprawa jest bardziej skomplikowana przy drużynie wieloosobowej. Niezbyt szczęśliwe rozwiązanie mamy w grze Soccer. Tu możemy kierować każdym z zawodników, wybór dokonywany jest wciskaniem FIRE. Jeszcze gorzej jest w Match Day - - kierujemy tym, który jest najbliżej piłki. W rezultacie można dostać oczopląsu i często w ogóle nie wiadomo, co się dzieje.

Nie ma co mówić o grze One on One — każdy steruje jednym koszykarzem. Warto natomiast zatrzymać się nad World Series Basketball — koszykówką z prawdziwego zdarzenia, z zachowaniem wszystkich obowiązujących przepisów. Przypomnę pokrótce te, których zazwyczaj nie znamy.

Przede wszystkim, po przejściu z piłką na połowę przeciwnika, piłka nie może być cofnięta z powrotem na swoją część boiska. Nie może też być w posiadaniu jednej drużyny dłużej, niż 30 sekund. Napastnik nie może przebywać z piłką pod koszem przeciwnika dłużej, niż 3 sekundy. Dotknięcie ręki przeciwnika powoduje zaliczenie jednego punktu foulowego; po zdobyciu czterech takich punktów drużyna przeciwna wykonuje rzuty osobiste. Następne rzuty przyznawane są za każdy dodatkowy punkt foulowy. Za każdy celny rzut drużyna dostaje jeden punkt, za rzut do kosza w trakcie gry dwa punkty, za rzut zza wyznaczonego pola trzy punkty.

Gra Basketball jest bardzo elastyczna w użyciu. Można wybrać jeden z sześciu poziomów trudności lub praktykę — po boisku biega wtedy tylko jedna drużyna. Do wyboru jest również długość połowy meczu — 3, 6 lub 9 minut. Sterowanie odbywa się dowolnym joystickiem lub zdefiniowanymi klawiszami. Można również ustalić kolor boiska jako jeden z ośmiu. Gracze automatycznie stają się kontrastowi.

Dlaczego World Series? Gra przewidziana jest jako rozgrywki w skali światowej. Oczywiście przed meczem podaje się nazwę swojej drużyny. Jeśli gra się z komputerem, po pierwszej przegranej następuje dyskwalifikacja. Gdy mecz jest wygrany, jego wynik zostaje zapamiętany w tabeli, a drużyna przechodzi szczebel wyżej. Nie ma co mówić o zdobyciu mistrzostwa świata, sukcesem jest wygrać mecz na najniższym szczeblu!

Firma: Imagine Software
Komputer: ZX Spectrum 48 /+, Amstrad/ Schneider

S.O.S.

Mam komputer Atari 65XE. Szukam gier: RAMBO, GUN FRIGHT, TRASHMAN i ARKANOID w wersji kasetowej.
Jacek Kuznicz ul. ZMW 7a m 29 75-342 Koszalin

Czy ktoś może zamienić się ze mną na gry: AVENGER, THE WAY OF THE TIGER, ASTERIX and THE MAGIC CAULDRON, RAMBO, ZORRO w wersji na Spectrum+?
Jakub Sandecki ul. Jarocińska 1/215
04-171 Warszawa

Pilnie szukam gry BOMB JACK na Spectrum. Odkupię lub wymienię na inne.
Leszek Tamarzewski ul. Stroma 1 59-820 Leśna woj. jeleniogórskie

Poszukuję dokładnych map, a także nieśmiertelności i innych ułatwień do gier: AVENGER, STRIKE FORCE COBRA, SABOTEUR II, RAMBO II na Spectrum. W zamian inne.
Piotr Szymanski ul. Targowa 15 m 63
03-727 Warszawa

Nie wiem, co zrobić, gdy się stoi przed murkiem w grze THE NINJA MASTER. Szukam opisów do gier: V, FYLER FOX, THE ARC OF YESOD. W zamian inne.
Hubert Czerski ul. Janinówka 14 m 10
03-562 Warszawa

Jak uzyskać nieśmiertelność w grach: COMMANDO, COBRA STALLONE, CHUCKIE EGG, BARBARIAN? Co zrobić w grze LIVINGSTONE?
Krzysztof Procek ul. Nadgórników 4 m 10
40-206 Katowice

Chodzi mi o sposób grania w KENNEDY APPROACH. Szukam też opisu do gry SABOTEUR I, II i III. Komputer Atari. W zamian inne.
Patryk Siewierski ul. Jagiellońska 6a/5
70-436 Szczecin

Poszukuję instrukcji do gry GUNLAW (RAMBO) i GHOSTBUSTERS na Atari 65 XE. W zamian inne.
Andrzej Ragus os. Życzyn 08-451 Życzyn

Mamy trudności z grą ZORRO na Atari 800 XE. Nie możemy wyjść z grobowca. Będziemy wdzięczni za pomoc.
Marcin i Tamara Korczak ul. Polna 46 m 5
00-644 Warszawa

Poszukuję następujących gier na Atari 65 XE: BARBARIAN, POP EYE, WINTER GAMES, ARKANOID, NINJA, SILENT SERVICE, PANAMA JOE i innych. W zamian 20 gier i programów użytkowych
Marek Jaśkowski ul. Matejki 25/4 87-100 Toruń

POKErzysta SPECTRUM

Poke'i do wpisania w loadery lub na programie COPY-COPY (najczęściej w główny segment).

DEFENDA	— POKE 37530,52 : POKE 37283,0
ASTRO BLASTER	— POKE 27422,0 . POKE 26396,255
SWEEVO'S WORLD	— POKE 33219,0
MUTANT MONTY	— POKE 54959,0
KNIGHT TYME	— POKE 24584,255 : POKE 24585,255 . POKE 45322,255 : POKE 45323,255 : POKE 41456,0 : POKE 41457,0
SCOOBY DOO	— POKE 64027,86 : POKE 64028,5 lub POKE 29614,0
TERMINUS	— POKE 45583,0 : POKE 47023,0
ROBIN OF THE WOOD	— POKE 49911,0
MARBLE MADNESS	— POKE 38579,0

Całe loadery do przepisania i zgrania na taśmę (zgrania na taśmę komendą GO TO 1987). Stary loader należy pominąć.

JUDGE DREDD — MELBOURNE HOUSE

```
10 REM ...M.1...
20 CLEAR VAL „24700"
30 LOAD "" CODE 16384 : LOAD "" CODE
40 POKE 24963,24
50 RANDOMIZE USR VAL "24736"
1987 SAVE CHR $ 22 + CHR $ 1 + CHR $ 0 + "DREDD"
+CHR $ 6 LINE 0
```

Loader do oryginału gry URIDIUM — HEWSON CONSULTANS

```
10 CLEAR 62719 : RESTORE : REM ...M.1...
20 LET W=2 : LET T=0
30 FOR I=23296 TO 23364
40 READ A : LET T=T+W*A : POKE I,A : LET W=W+1
50 NEXT I
60 IF T<>24746 THEN PRINT "GDZIEŚ JEST BŁĄD"
   BEEP 1,1 . STOP
70 POKE 23353,0 : POKE 23558,0
80 PRINT AT 21,5 : "START URIDIUM"
90 RANDOMIZE USR 23296
500 DATA 49, 0, 92, 221, 33
501 DATA 0, 0, 17, 125, 2
502 DATA 62, 255, 55, 205, 86
503 DATA 5, 48, 238, 221, 33
504 DATA 0, 64, 17, 0, 27
505 DATA 62, 255, 55, 205, 86
506 DATA 5, 221, 33, 0, 0
507 DATA 17, 0, 1, 62, 255
508 DATA 243, 205, 169, 5, 221
509 DATA 33, 0, 92, 17, 0
510 DATA 164, 62, 255, 205, 169
511 DATA 5, 62, 1, 50, 168
512 DATA 152, 62, 2, 50, 147
513 DATA 53, 195, 80, 253
1987 SAVE CHR $ 22 + CHR $ 1 + CHR $ 0 + "URI-
DIUM" LINE 0
```

Loader do oryginału gry FIRELORD — HEWSON CONSULTANS

```
10 CLEAR (65535 : REM ...M.1...
20 LET AA=USR "a"
30 READ N
40 IF N=999 THEN GO TO 60
50 POKE AA,N : LET AA=AA+1 : GO TO 30
60 PAPER 0 : INK 0 : BORDER 0 : BRIGHT 0 : CLS
70 RANDOMIZE USR 65368
510 DATA 62, 255, 55, 221, 33, 39
511 DATA 244, 17, 125, 2, 205, 86
512 DATA 5, 48, 243, 62, 255, 55
513 DATA 221, 33, 0, 64, 17, 87
514 DATA 191, 205, 86, 5
515 DATA 175, 50, 205, 134 -nieśmiertelny
516 DATA 175, 50, 205, 125, 135 — ułatwienia w handlu
517 DATA 175, 50, 170, 150, 62, 7, 50, 156, 150
518 DATA 62, 58, 50, 168, 136
519 DATA 175, 50, 38, 156, 62, 58, 50, 67, 156
520 DATA 195, 79, 94, 999
1987 SAVE "CHR $ 22 + CHR $ 1 + CHR $ 0 + "LORD"
+ CHR $ 6 LINE 0
```

M.1

PRZEDSTAWIAMY WARSAW BASIC (9)

WYWOŁANIE PROCEDURY (2)

Od redakcji: Zdarzyła nam się okropna gafa — w numerze 10/88 pominęliśmy dziewiąty odcinek Warsaw Basica. Naprawiając nasz błąd przepraszamy jednocześnie Autorów i Czytelników.

Wywołanie procedury w Warsaw BASIC'u realizuje instrukcja CALL. Do wywołania procedury z dyskietki wystarczy podanie jej nazwy i listy parametrów. W pętli interpretera dokonuje się sprawdzenia składni wywołania i jeśli listy parametrów poprzedzonych nazwą nie można zakwalifikować jako zmiennej tablicowej, to indukowane jest wywołanie CALL.

CALL wykonuje szereg czynności związanych z wywołaniem, a mianowicie:

(1) sprawdzenie czy procedura jest w pamięci operacyjnej bezpośrednio za zbiorem zmiennych programu wywołującego,
(2) sprawdzenie czy procedura jest w RAM-dysku,
(3) utworzenie nowego poziomu,
(4) sprawdzenie czy procedura jest w pamięci zewnętrznej,
(5) wczytanie procedury do komputera,
(6) przekazanie parametrów,
(7) poinformowanie, które z parametrów są przekazywane przez nazwę,
(8) przekazanie sterowania procedury.

Objętość CALL z Warsaw BASIC'a zmusza nas do rozłożenia prezentacji tej instrukcji na kilka etapów. Ograniczymy się przy tym do przedstawienia czynności związanych z realizacją punktów (1), (3), (4), (5), (6) i (8). W tym odcinku prezentujemy program, który realizuje czynności (3), (4), (5) i (8).

Interpreter Warsaw BASIC'a w czasie realizacji punktu (4) proponuje wymianę nośnika, jeśli poszukiwanej procedury nie ma na dyskietce znajdującej się aktualnie w stacji dysków. W prezentowanym programie pominiemy to udogodnienie. Możliwe jest zastąpienie go odpowiednimi czynnościami operatorskimi w trybie bezpośrednim. Jeśli wywołanie procedury zakończy się błędem FILE NOT FOUND, to po powrocie na poziom wywołujący (£0) i wymianie dyskietki można wznowić poszukiwanie procedury poprzez skok za pomocą instrukcji GOTO do linii, w której występuje instrukcja wywołania.

Odpowiednikiem CALL z Warsaw BASIC'a jest w naszym programie instrukcja £C. Program prezentowany w tym odcinku pozwala na wykorzystanie instrukcji £C w następującym formacie:

£C „nazwa procedury", nr urządzenia

Zamiast parametru „nazwa procedury" w postaci stałej można użyć wyrażenia tekstowego. W przypadku wywoływania procedury zapisanej na taśmie magnetycznej numer urządzenia można pominąć. Instrukcja £C jest zrealizowana w liniach od adresu $C707 do adresu $C79B (por. program 1), w tym w liniach od adresu $C76D do adresu $C79B zrealizowano załadowanie procedury do pamięci operacyjnej.

Z instrukcją £C integralnie związana jest instrukcja £H, która realizuje funkcje instrukcji PROCEDURE Warsaw BASIC'a. £H pełni rolę nagłówka i musi być pierwszą instrukcją pierwszej linii procedury. Program prezentowany w tym odcinku pozwala na wykorzystanie instrukcji £H w następującym formacie:

£H „nazwa procedury"

Parametr „nazwa procedury" musi być stałą tekstową i musi pokrywać się z nazwą, pod którą procedura została zapisana w pamięci zewnętrznej. Przypominamy, że w Warsaw BASIC'u linia zawierająca nagłówek nie musi być pierwszą linią programu umieszczonego na poziomie wywoływanym. Taka organizacja umożliwia umieszczenie na tym poziomie kilku procedur połączonych w zestaw. Wywołanie procedury z zestawu powoduje, że wykonywanie procedury rozpoczyna się od jej nagłówka. Takie wywołanie musi być poprzedzone sprawdzeniem czy wywoływana procedura jest już w pamięci operacyjnej. Ponieważ w tym odcinku nie przewidujemy realizacji sprawdzania, to pozostaniemy przy wymaganiu rozpoczynania tekstu procedury od nagłówka. Instrukcja £H jest zrealizowana w liniach od adresu C79e do adresu C7b5 (por. program 1).

£C po wywołaniu procedury przekaże sterowanie instrukcji £H, po wykonaniu której będą sekwencyjnie wykonywane instrukcje zawarte w tekście. Wykonanie procedury powinno kończyć się przekazaniem sterowania do pierwszej instrukcji po instrukcji wywołania. Umożliwia to rozkaz £E, który realizuje funkcje £0 (przełączenie na poziom wywołujący) i RETURN (powrót do pierwszego rozkazu po £C). Instrukcja £E jest zrealizowana w liniach od adresu C7B8 do adresu C7C9 (por. program 1). £E tak jak RETURN w standardowym podprogramie wywoływanym przez GOSUB nie musi być ostatnią instrukcją w procedurze. Jest natomiast instrukcją, która jest wykonywana jako ostatnia.

W Warsaw BASIC'u wykonanie procedury można zakończyć jeszcze na dwa inne sposoby. PROC ERROR — kończy działanie procedury w przypadku nadzwyczajnym i powoduje, że interpreter po przerwaniu wykonania zgłosi się na poziomie wywołującym. PROC MOVE — kończy wykonanie procedury, powoduje przełączenie się do programu wywołującego z przeniesieniem pierwszej tablicy z procedury do programu wywołującego na ostatnie miejsce. PROC MOVE służy do tworzenia własnych typów danych w oparciu o typ tablicowy.

Przypominamy, że numery linii procedury mogą być takie same jak numery linii programu wywołującego. Zmienne używane w procedurze są zmiennymi lokalnymi i ich nazwy przesłaniają takie same nazwy zmiennych z programu wywołującego.

W prezentowanej realizacji wywołania nie ma jeszcze możliwości przekazywania parametrów do procedury, ale Czytelnicy, którzy dołączą program 2 do poprzednich części i uruchomią całość będą mogli korzystać z bezparametrowych procedur takich jak np. procedura kopiująca zawartość ekranu znakowej rozdzielczości na drukarce. Procedura ta nie potrzebuje parametrów, bowiem potrzebne dane są umieszczone w pamięci ekranu. Dostosowując istniejące programy do współpracy z naszym rozbudowanym interpreterem trzeba tylko pamiętać, że pamięć ekranu została przesunięta na strony od 12*16 do 15*16 włącznie. Po takim dostosowaniu program kopiujący ekran stanie się właściwie dodatkowym rozkazem BASIC'a i będzie go można wykorzystywać w każdym innym programie już bez potrzeby dokonywania w nim żadnych przeróbek.

Na zakończenie prezentujemy przykład programu wywołującego i procedury, które realizują te same funkcje co program i procedura z poprzedniego odcinka. Aby uruchomić ten program należy uprzednio wpisać do pamięci i nagrać procedurę „KASIA", w zbiorze o tej samej nazwie. Wtedy można wpisać program wywołujący i uruchomić go. Po wykonaniu dyrektywa £P pozwala wyświetlić tekst procedury (LIST). £0 pozwala wrócić na poziom 0 i ponownie uruchomić program (RUN). Ponowne uruchomienie spowoduje, że „KASIA" ponownie będzie wczytywana z pamięci zewnętrznej, mimo że jest już w komputerze. Aby tego uniknąć należy przed załadowaniem sprawdzić, czy procedura jest już w pamięci, ale o tym jak to zrealizować opowiemy w następnym odcinku.

Krzysztof Gajewski
Bogusław Radziszewski

Program 1

```
.. C445 06 C7
.. C449 B7 C7
.. C44F 9D c7

.. C6D6 8D FB CF STA $CFFB
.. C6D9 A9 00    LDA #$00
.. C6DB 20 BD FF JSR $FFBD
.. C6DE A2 01    LDX #$01
.. C6E0 A0 00    LDY #$00
.. C6E2 20 BA FF JSR $FFBA
.. C6E5 20 06 E2 JSR $E206
.. C6E8 20 57 E2 JSR $E257
.. C6EB 20 06 E2 JSR $E206
.. C6EE 20 00 E2 JSR $E200
.. C6F1 A0 00    LDY #$00
.. C6F3 86 49    STX $49
.. C6F5 4C BA FF JMP $FFBA

.. C6F8 48       PHA
.. C6F9 AC CA C4 LDY $C4CA
.. C6FC B9 F0 CF LDA $CFF0.Y
.. C6FF C9 02    CMP #$02
.. C701 F0 83    BEQ $C686
.. C703 68       PLA
.. C704 4C 7B C6 JMP $C67B

.. C707 A9 08    LDA #$08
.. C709 20 F8 C6 JSR $C6F8
.. C70C 20 D6 C6 JSR $C6D6
.. C70F A5 7A    LDA $7A
.. C711 48       PHA
.. C712 A5 7B    LDA $7B
.. C714 48       PHA
.. C715 20 CC C7 JSR $C7CC
.. C718 F0 0F    BEQ $C729
.. C71A 20 A6 C5 JSR $C5A6
.. C71D 20 80 C5 JSR $C580
.. C720 20 6D C7 JSR $C76D
.. C723 20 95 C6 JSR $C695
.. C726 20 C2 C5 JSR $C5C2
.. C729 68       PLA
.. C72A 85 7B    STA $7B
.. C72C 68       PLA
.. C72D 85 7A    STA $7A
.. C72F A0 00    LDY #$00
.. C731 B1 7A    LDA ($7A).Y
.. C733 C9 2C    CMP #$2C
.. C735 D0 03    BNE $C73A
.. C737 20 CF C7 JSR $C7CF
.. C73A A5 14    LDA $14
.. C73C A4 15    LDY $15
.. C73E 85 FB    STA $FB
.. C740 84 FC    STY $FC
.. C742 A0 03    LDY #$03
.. C744 B1 31    LDA ($31).Y
.. C746 85 14    STA $14
.. C748 C8       INY
.. C749 B1 31    LDA ($31).Y
.. C74B 85 15    STA $15
.. C74D A9 03    LDA #$03
.. C74F 20 FB A3 JSR $A3FB
.. C752 20 80 C5 JSR $C580
.. C755 A5 7B    LDA $7B
.. C757 48       PHA
.. C758 A5 7A    LDA $7A
.. C75A 48       PHA
.. C75B A5 3A    LDA $3A
.. C75D 48       PHA
.. C75E A5 39    LDA $39
.. C760 48       PHA
.. C761 A9 8D    LDA #$8D
.. C763 48       PHA
.. C764 20 09 A9 JSR $A909
.. C767 20 BC A8 JSR $A8BC
.. C76A 4C AE A7 JMP $A7AE

.. C76D A9 00    LDA #$00
.. C76F 85 B9    STA $B9
.. C771 85 0A    STA $0A
.. C773 A6 2B    LDX $2B
.. C775 A4 2C    LDY $2C
.. C777 20 D5 FF JSR $FFD5
.. C77A B0 1F    BCS $C79B
.. C77C 20 B7 FF JSR $FFB7
.. C77F 8A       TXA
.. C780 48       PHA
.. C781 98       TYA
.. C782 48       PHA
.. C783 AD FB CF LDA $CFFB
.. C786 C9 08    CMP #$08
.. C788 D0 0C    BNE $C796
.. C78A 68       PLA
.. C78B 85 2E    STA $2E
.. C78D 68       PLA
.. C78E 85 2D    STA $2D
.. C790 20 8E A6 JSR $A68E
.. C793 4C 33 A5 JMP $A533
.. C796 68       PLA
.. C797 68       PLA
.. C798 18       CLC
.. C799 90 F5    BCC $C790
.. C79B 4C D1 E1 JMP $E1D1

.. C79E 20 1D A8 JSR $A81D
.. C7A1 20 73 00 JSR $0073
.. C7A4 C9 22    CMP #$22
.. C7A6 D0 F9    BNE $C7A1
.. C7A8 20 73 00 JSR $0073
.. C7AB F0 25    BEQ $C7D2
.. C7AD C9 3A    CMP #$3A
.. C7AF F0 21    BEQ $C7D2
.. C7B1 C9 2C    CMP #$2C
.. C7B3 D0 EC    BNE $C7A1
.. C7B5 4C D5 C7 JMP $C7D5

.. C7B8 20 95 C6 JSR $C695
.. C7BB 20 BF C5 JSR $C5BF
.. C7BE A5 FB    LDA $FB
.. C7C0 A4 FC    LDY $FC
.. C7C2 85 41    STA $41
.. C7C4 84 42    STY $42
.. C7C6 20 73 00 JSR $0073
.. C7C9 4C D2 A8 JMP $A8D2

.. C7CC A9 01    LDA #$01
.. C7CE 60       RTS

.. C7CF A9 01    LDA #$01
.. C7D1 60       RTS

.. C7D2 4C B6 C5 JMP $C5B6

.. C7D5 A9 01    LDA #$01
.. C7D7 60       RTS
```

Program 2

```
600 PRINT"Część 6"
602 X=50902: N=257: C=0
604 FOR I=0 TO N: READA: POKEX+I.A: C=C+A: NEXT
606 IF C=31946 THEN PRINT"Część 6 OK": GOTO610
608 PRINT"Błąd w części 6": END
610 Y=50245: Z=199
612 POKEY+1.Z: POKEY+5.Z: POKEY+11.Z
614 POKEY.6: POKEY+4.183: POKEY+10.157
616 Y=53248: POKEY-3.160: POKEY-2.207
618 DATA 141.251.207.169.0.32.189.255.162.1.160.0
620 DATA 32.186.255.32.6.226.32.87.226.32.6.226
622 DATA 32.0.226.160.0.134.73.76.186.255.72.172
624 DATA 202.196.185.240.207.201.2.240.131.104.76.123
626 DATA 198.169.8.32.248.198.32.214.198.165.122.72
628 DATA 165.123.72.32.204.199.240.15.32.166.197.32
630 DATA 128.197.32.109.199.32.149.198.32.194.197.104
632 DATA 133.123.104.133.122.160.0.177.122.201.44.208
634 DATA 3.32.207.199.165.20.164.21.133.251.132.252
636 DATA 160.3.177.49.133.20.200.177.49.133.21.169
638 DATA 3.32.251.163.32.128.197.165.123.72.165.122
640 DATA 72.165.58.72.165.57.72.169.141.72.32.9
642 DATA 169.32.188.168.76.174.167.169.0.133.185.133
644 DATA 10.166.43.164.44.32.213.255.176.31.32.183
646 DATA 255.138.72.152.72.173.251.207.201.8.208.12
648 DATA 104.133.46.104.133.45.32.142.166.76.51.165
650 DATA 104.104.24.144.245.76.209.225.32.29.168.32
652 DATA 115.0.201.34.208.249.32.115.0.240.37.201
654 DATA 58.240.33.201.44.208.236.76.213.199.32.149
656 DATA 198.32.191.197.165.251.164.252.133.65.132.66
658 DATA 32.115.0.76.210.168.169.1.96.169.1.96
660 DATA 76.182.197.169.1.96
```

ZAPIS LICZB W RAM ZX SPECTRUM (1)

Spośród zapisu wszystkich danych programu w pamięci RAM, zapis liczb jest zapisem w szczególny sposób wyróżnionym, gdyż każda liczba oprócz tego, że zapisana jest zawsze w dwóch miejscach pamięci, a jeśli stanowi argument zmiennej, to dodatkowo w trzecim, jest zapisywana w sposób uzależniony od jej wartości i postaci jak też i od miejsca zapisu w pamięci.

Zasadniczo można wyróżnić trzy następujące sposoby zapisu każdej liczby:

a) zapis w postaci ciągu kodów poszczególnych znaków cyfr i symboli liczby zwany niekiedy zapisem ekranowym,
b) zapis binarny albo tzw. zapis uproszczony,
c) zapis zmiennoprzecinkowy nazywany także zapisem zmiennopozycyjnym.

Każda liczba zapisana jest zawsze w postaci a), natomiast drugi i ewentualnie trzeci zapis może być albo w postaci b) albo w postaci c).

Sposób a) jest identyczny dla każdej liczby i stosowany wyłącznie w pamięci programu. Na jego podstawie wypisywana jest liczba na ekranie przy listingu programu. Sposób ten polega na tym, że w komórkach pamięci zapisywane są kody wszystkich cyfr, znaków i ewentualnych symboli, z których składa się liczba w takiej kolejności, w jakiej te cyfry, symbole i znaki były wyprowadzone z klawiatury. Liczba zapisywana tym sposobem zajmuje tyle komórek pamięci, ile znaków, cyfr i symboli posiada dana liczba. Np. liczba **1.2E+26** będzie w pamięci zapisana jako ciąg kodów **1**, **.**, **2**, **E**, **+**, **2**, **6**, czyli 49–46–50–69–43–50–54. Ten sposób zapisu jest oczywisty i wydaje się, że po tych kilku uwagach na jego temat nie powinien budzić wątpliwości.

Sposób b) i c) stosowany jest zarówno w pamięci programu jak też i w polu zmiennych. Zajmuje on zawsze pięć komórek pamięci, które w polu programu znajdują się bezpośrednio za komórką pamięci z liczbą „14" oddzielającą obszar zapisu ekranowego od obszaru 5-cio bajtowego. Liczba „14" jest więc niejako granicą sygnalizującą koniec postaci ekranowej i początek postaci binarnej danej liczby (zapis b)) lub zmiennoprzecinkowej (zapis c)). W polu zmiennych te pięć komórek przyporządkowanych danej liczbie znajduje się w miejscu wynikającym z kolejności występowania zmiennych w programie i poprzedzone są dodatkowymi informacjami o rodzaju zmiennej.

Jedynym wskaźnikiem obszaru pamięci, w którym znajduje się te pięć komórek, jest adres początku pola zmiennych przechowywany w zmiennej systemowej VARS (komórka 23627 — młodszy bajt i komórka 23628 — starszy bajt tego adresu).

Nie wnikając w szczegóły, które zostaną omówione niżej, należy zaznaczyć, że zapis liczby w postaci b) lub w postaci c) jest identyczny w polu pamięci programu i w pamięci pola zmiennych jedynie dla liczb dodatnich, dla liczb ujemnych natomiast zapis ten, jak zobaczymy, różni się w sposób zasadniczy.

Zapis w postaci uproszczonej

Zapis ten stosowany przez system dla liczb całkowitych z przedziału **−65535** do **+65535** realizowany jest przy pomocy tzw. starszego albo bardziej znaczącego bajtu obliczanego wg równania:

$$b_s = INT\ (a/256) \qquad (1)$$

gdzie: a — dana liczba
b_s — bajt starszy

i zapisanego w czwartej komórce pamięci 5-cio bajtowego formatu oraz tzw. młodszego albo mniej znaczącego bajtu obliczanego wg równania:

$$b_m = a - 256 * b_s \qquad (2)$$

gdzie: b_m — bajt młodszy

i zapisanego w trzeciej komórce

W pamięci programu brak jest informacji o znaku liczby, w związku z czym całkowite dodatnie i ujemne z w/w zakresu zapisywane są tu identycznie. Np. liczba **−384** i **+384** w 5-ciu bajtach pamięci programu będzie miała tę samą postać tzn. 0–0–−128–1–0 bo 384 = 128 + 1 * 256. Informacja o znaku znajduje się dopiero w 5-ciu bajtach pamięci pola zmiennych. Tutaj liczba dodatnia zapisana jest identycznie jak w polu pamięci programu, natomiast liczba ujemna, mimo że zapisana jest również przy użyciu starszego i młodszego bajtu to jednak bajty te nie reprezentują tej liczby lecz jej uzupełnienie do dwóch w stosunku do liczby 65536. Uzupełnieniem do dwóch przykładowej liczby „**−384**" w stosunku do liczby 65536 będzie liczba **c = 65536 − 384 = 65152.**

Bajt starszy będzie więc:

$$b_s = INT\ (65152/256) = 254$$

Bajt młodszy natomiast będzie:

$$b_m = c - 256 * b_s = 65152 - 256 * 254 = 128$$

Ponadto, przy zapisie tej liczby do drugiego bajtu wpisywana jest dodatkowo liczba 255 informująca, że zapisana liczba jest liczbą ujemną.

Zapis przykładowej liczby **−384** w 5-ciu bajtach pola zmiennych będzie miał zatem postać: 0–255–−128–254–0. Dla liczby **+384**, jak wiemy, zapis był 0–0–128–1–0.

Widzimy więc, że zapis liczby o tej samej bezwzględnej wartości lecz opatrzonej różnym znakiem, w pamięci pola zmiennych jest zasadniczo różny. Jeżeli jednak uzmysłowimy sobie, że tak naprawdę to przecież dana liczba ze znakiem „+" i ta sama ze znakiem „−", to na osi liczbowej dwie różne liczby oraz to, że operacje arytmetyczne na liczbach dodatnich i liczbach ujemnych zapisanych w kodzie uzupełnienia do dwóch są logicznie prostsze, to mając jeszcze na uwadze, że logika systemu komputerowego to logika prawie absolutna, różnica ta nie powinna nas już dziwić.

Zapis w postaci zmiennoprzecinkowej

Zapis ten stosowany jest dla liczb ułamkowych z całego zakresu liczb, na których system komputera ZX Spectrum może prowadzić obliczenia, tj. od −1.70141183469(9)E+38 do +1.70141183469(9)E+38 z wyłączeniem liczb większych od −2.938735877 E−39 lecz mniejszych od +2.938735877 E−39 (liczby te, jak wiemy, system traktuje jako liczby równe 0) oraz liczb całkowitych mniejszych od +65535 i większych od −65535.

Ten sposób zapisu bazuje na znanym nam fundamentalnym równaniu w postaci:

$$a = m * 2^n \qquad (3)$$

Aby system komputera mógł przy użyciu tego równania zapisać liczbę w komórkach pamięci, musi dokonać potrzebnych przeliczeń celem „przygotowania" danych dla poszczególnych komórek 5-cio bajtowego formatu.

Czyni to w następujący sposób:

1. Oblicza wykładnik potęgowy wg równania:

$$n = INT\left(\frac{\ln /2*a/}{\ln 2}\right) \qquad (4)$$

Równanie to otrzymano z obustronnego zlogarytmowania równania (3) dla m = 0.5

2. Oblicza wyrażenie:

$$K = 2^n \qquad (5)$$

3. Oblicza mantysę „m" wg równania

$$m = \frac{a}{K} \qquad (6)$$

UWAGA!

Mantysa jest zawsze większa lub co najmniej równa 0.5, 0.5 dziesiętnie to 0.1 binarnie. Jeżeli zatem każda mantysa posiada składnik 0.5 dziesiętnie, czyli binarnie 0.1, to znaczy, że w zapisie binarnym mantysy pierwszym znakiem po przecinku jest zawsze „1". Ta właściwość mantysy pozwala uwolnić pierwszy bit po przecinku i wykorzystać go do informacji o znaku liczby, natomiast stałą wartość 0.5 uwzględniać wyłącznie przy ustalaniu wartości liczby. Nie jest to żadne fałszerstwo gdyż w pamięci system zapisuje liczby dla swoich potrzeb a nie dla programisty. Ten natomiast, jeżeli poprosi o liczbę, to system w mantysie uwzględni składnik 0.5 i na ekranie czy też do obliczeń poda liczbę o właściwej wartości.

4. Z tak obliczonej mantysy odrzuca składnik 0.5 a pozostałą wartość zamienia na 32-bitową postać binarną.
5. 32-bitową mantysę z zerem po przecinku dzieli na cztery równe części po osiem bitów. Mantysa nie zawsze musi zawierać wszystkie wyrazy szeregu nieskończonego, w związku z czym rozwinięcie na postać binarną może zawierać mniej niż 32 bity znaczące. W takim przypadku tworzone są trzy, dwa, jeden bajt znaczący a jeżeli mantysa m = 0.5, wówczas wszystkie cztery bajty są równe 0.
6. Tak przygotowane bajty wpisuje kolejno do czterech komórek pamięci 5-cio bajtowego formatu zapisu liczby poczynając od komórki drugiej. W tej właśnie komórce ósmy bit „jest wolny". W bicie tym system wpisuje informację o znaku liczby (1 — liczba ujemna, 0 — liczba dodatnia) ale czyni to tylko w 5-cio bajtowym formacie pola zmiennych. W identycznych 5-ciu komórkach w pamięci programu informacja o znaku nie istnieje, chociaż sam zapis liczby realizowany jest w taki sam sposób i ósmy bit też „jest wolny". Znajduje się tu jednak zawsze liczba 0. Stąd wniosek, że w tym bajcie w polu programu największą zapisaną liczbą może być liczba 127 (7 jedynek), niezależnie od tego czy liczba jest dodatnia czy też ujemna, w polu zmiennych natomiast w przypadku liczby dodatniej będzie także liczba 127 ale przy ujemnej będzie 255 (8 jedynek), przy czym ósma odczytywana jest jednak jako informacja o znaku „−".
7. Do pierwszej komórki 5-cio bajtowego formatu wpisuje liczbę ustaloną wg równania:

$$n_1 = 128 + n \qquad (7)$$

gdzie: n — wykładnik potęgowy obliczony na podstawie równania (4)

Zauważmy, że tak ustalona liczba i umieszczona w tej komórce pozwala zapisać wykładnik o wartościach od n = −128 (n_1 = 0) do n = + 127 (n_1 = 255), dlatego też $2^{-128} = 2.938735877*10^{-39}$ jest najmniejszą, a $2^{+127} = 1.70141183469*10^{+38}$ największą liczbą, na której system ZX Spectrum może prowadzić obliczenia. Powyższe stwierdzenie jest formalnie prawdziwe ale zapis $n_1 = 0$ z punktu widzenia logiki systemu jest wątpliwy, gdyż niezależnie od tego, czy do komórki pamięci zostaje zapisane zero czy nie, to ono tam jest, dlatego też, aby tę wątpliwość wykluczyć, najmniejsza liczba jaka do tej komórki jest wpisywana wynosi n_1 = 1, co odpowiada wykładnikowi n = −127. W tej sytuacji najmniejsza liczba a_m wynika teraz z równania (3) i określona jest dla m = 0.5 (najmniejsza) i dla n = −127 (najmniejsza) czyli:

$$a_m = m*2^n = 0.5*2^{-127} \tfrac{1}{2}\ 2^{-127} = 2^{-128} = 2.938735877E + 38$$

Podobnie liczba największa jest określona dla m = 0.99999999 (największa) i n = +127 (największy), czyli:

$$a_w = m \cdot 2^n = 0.99999999*2^{+127} = 1.70141183469E + 38$$

Za miesiąc — przykłady

Piotr Sumara

ZACZĘŁO SIĘ OD SPECTRUM

Nie jest to bajka ani reklama. Poniższy zarys historii pracowni informatycznej w II L.O. im. Stefana Batorego w Warszawie dedykuję młodym entuzjastom informatyki w polskich szkołach oraz ich nauczycielom — tym, którym już się udało i tym, którzy mają to jeszcze przed sobą.

Dawno temu, gdy tylko nieliczni mieli za sobą pierwszy kontakt z mikrokomputerem (głównie w Pracowni Podstaw Informatyki Pałacu Młodzieży), a konkretniej na jesieni 1984 r., rozpoczęły się spotkania Koła Informatycznego. Zainteresowanie było bardzo małe. Niemal każdy zapytany o tajemniczy anons na tablicy ogłoszeń lekceważąco machał ręką — jedni nie wiedzieli co kryje się pod tą nazwą, drudzy — tych było najmniej — już tak zagłębili się w problematykę mikrokomputerową, że nie interesowały ich podstawy. Jeszcze inni, tych było najwięcej, uważali zajęcia „na sucho" za stratę czasu. Wiemy przecież, że nawet bardzo cierpliwych słuchaczy nie mógł zadowolić wykład nauczyciela nie poparty żadnymi praktycznymi przykładami. Należy jednak zaznaczyć, że pomimo braku sprzętu główny opiekun Koła, mgr Witold Kranas robił co mógł. Dzięki Jego staraniom zajęcia raz w tygodniu odbywały się w Centrum Astronomicznym PAN przy ulicy Bartyckiej, gdzie oczywiście można było korzystać z mikrokomputera. Jednocześnie Dyrekcja Szkoły trzymała rękę na pulsie i gdy tylko władze oświatowe otrzymały dar w postaci kilku komputerów, wystąpiono z prośbą o przydzielenie jednego z nich. Cóż to była za radość!

ZX SPECTRUM 16 KB na ok. 800 uczniów...
No, może nie na tylu, bo choć zainteresowanie Kołem znacznie wzrosło, to jego liczebność porównywalna była zaledwie z jedną klasą. Ale apetyt tej grupy tylko wzrósł, bo co można zobaczyć na ekranie telewizora z końca dużej sali...

Dla nauczycieli fizyki prowadzących zajęcia Koła stało się jasne, że uczniowie tej szkoły nie chcą być „głupsi" od swych rówieśników. Dyrektor Szkoły, mgr Teresa Garncarzyk uznała, że nie mogą i z funduszy szkoły zakupiono drugi komputer.

Po rozszerzeniu pamięci pierwszego, Koło rozpoczynało rok szkolny 1985/86 z dwoma ZX SPECTRUM 48 KB. Teraz zaczęła się zabawa — dziesiątki godzin spędzonych na zapleczu jednej z pracowni fizycznych, gdzie znajdował się sprzęt komputerowy.

By umożliwić uczniom swobodne, samodzielne korzystanie z wyposażenia zaplecza, mgr Witold Kranas wprowadził „legitymacje użytkownika ZX Spectrum". Ten zaszczytny tytuł wraz z dokumentem otrzymywało się po zdaniu egzaminu z obsługi sprzętu i znajomości języka BASIC. Z kronikarskiego obowiązku dodam, że pierwszymi, którym się to udało, byli uczniowie klasy drugiej.

„Użytkowników" przybywało, a mikrokomputerów nie. Czy liceum mogło nie reagować na projekt wprowadzenia informatyki jako przedmiotu szkolnego? Dla Dyrekcji i opiekunów Koła odpowiedź była oczywista. Do pamiętnego września 1986 roku przygotowano się więc solidnie: 10 nowych ZX Spectrum 48 KB z monitorami i magnetofonami oraz wymagane w programie nauczania LOGO.

Charakterystyczny szum Spectrum współpracującego z „Kasprzakiem MK 232" niósł się teraz echem po całej szkole. Pracownia, jedna z pierwszych w warszawskich liceach, powstała za ok. 2 miliony złotych, dla uczniów stała się czymś zupełnie normalnym i codziennym. „Kuć żelazo póki gorące" (o ile to możliwe, a w przypadku II L.O. było to możliwe) — dziś zajęcia informatyki odbywają się nawet w klasie o profilu biologiczno-chemicznym.

Nie będę opisywał dzień po dniu zabiegów i starań Pana mgr Kranasa — dawni „użytkownicy" nie mogli narzekać na nudę w pracowni. Jeszcze w tym samym roku szkolnym zawierała ona obok 12 Spectrum z monitorami i magnetofonami, dwie stacje dysków 3" Timex (do ZX Spectrum) oraz mikrokomputer AMSTRAD CPC 6128 z drukarką DMP 2000. Wszystkie urządzenia wykorzystywane były do tego stopnia, że przestawał istnieć problem ogrzewania wyziębionej sali. Zapalczywi entuzjaści zmuszali niestety swego nauczyciela do częstych odwiedzin serwisu. Dzięki firmie AUDIO-TRONIC, ulubione przez uczniów „spektrumny" nie traciły na zawsze swych sprzętowych walorów.

Zaczęto jednak dostrzegać inne braki, bowiem program podstaw informatyki w zasadzie byłby możliwy do zrealizowania, ale mając Spectrum trudno jest przekonać podopiecznych, że może być coś znacznie ciekawszego niż gry. Pan W. Kranas kształcąc już całe klasy przyszłych użytkowników komputerów, by pokazać im np. edytor tekstu lub PASCAL, potrzebował czegoś więcej.

O fundusze było już łatwiej, tak więc pojawił się pierwszy PC XT-Bondwell 8. W niedługim czasie zgromadzono do niego około 70 dyskietek.

W roku szkolnym 1987/88, w pracowni znajdował się również pożyczony szkole klon PC XT, który ostatecznie rozbudził ambicje uczniów i nauczycieli. Na tym komputerze przez cały rok wielu z nich (także absolwentów, którzy nagle przypomnieli sobie o starych murach) nabierało wprawy w obcowaniu z DOS-em. Później już władze oświatowe nie ociągały się z asygnowaniem milionowych sum. Zakupiono najpierw AMSTRAD-a PC 1512 DD, a następnie trzy podobne komputery firmy Schneider (wszystkie z jednym napędem dysków, jeden z twardym dyskiem). Imponujący to potencjał, ale nie dający jeszcze pełnej możliwości upowszechniania PC XT wśród uczniów.

Na nadchodzący rok planowane jest więc dopełnienie zestawu do poziomu: 10 komputerów w prostej konfiguracji (jednostka centralna 512-640 KB RAM, karta CGA, jeden napęd dysków elastycznych, monitor) oraz jeden twardy dysk i drukarka typu SG-15. Wizja sieci PC i pracy w TURBO PASCAL-u całymi klasami już w następnym roku ma wielkie szanse, by stać się rzeczywistością. Dziś z rozrzewnieniem wspomina się w Szkole SPECTRUM 16...

II Liceum Ogólnokształcące w Warszawie to przykład, dlatego proszę, aby nasi Czytelnicy nie pisali sprostowań o tym, że kluby komputerowe istniały już w roku 1982, a w jakiejś szkole uczniowie mają do dyspozycji 20 IBM PC AT z twardymi dyskami. To, w jaki sposób oceniamy karierę pracowni informatycznej w opisanej szkole, niech będzie dla nas samych miarą zaawansowania we wprowadzaniu informatyki do polskiej oświaty.

Piotr Bernatek

CZYTANIE DANYCH Z POWTÓRZENIAMI

Czasami zdarza się, że dane do obliczeń zawierają długie ciągi takich samych elementów. Aby uniknąć wielokrotnego wprowadzania z klawiatury tych samych liczb warto w takim przypadku pisać dane w specjalny sposób. Czytane dane można umieścić w zmiennej tekstowej, liczbę która się powtarza kończymy znakiem mnożenia, a po nim piszemy liczbę powtórzeń.
Zatem dane do obliczeń:
3 1 1 1 1 1 1 4 – 1 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 4
można napisać tak
3 1•6 4 – 1 2•7 0•8 4
Program mający czytać dane w ten sposób powinien zawierać opis następującego podprogramu czytania:

```
9000 REM Podprogram czytania danych z powtórzeniami
9010 REM Janusz Sz. 1987
9020 IF lp>1 THEN LET lp=lp-1 RETURN
9030 DIM a$(15)
9040 INPUT „l = " LINE a$(1 TO 15)
9050 LET i$=„1": LET ls=0: LET Poz=16
9060 FOR z=1 TO 15
9070 IF a$(z)=„•" THEN LET poz=z
9080 NEXT z
9090 LET li=VAL a$(1TO poz-1)
9100 IF poz <>16 THEN LET i$=a$(poz+1 TO 15)
9110 LET lp=VAL i$
9120 RETURN
```

W powyższym pdprogramie zmienna Li zawiera aktualnie czytaną liczbę, a zmienna Lp zawiera liczbę powtórzeń.
UWAGA: PRZED PIERWSZYM WYWOŁANIEM PODPROGRAMU NALEŻY BEZWZGLĘDNIE POD ZMIENNĄ Lp PODSTAWIĆ WARTOŚĆ 1.
A oto przykład programu wykorzystujący podprogram czytania:

```
10 LET lp=1:DIM a(5,5)
20 FOR i=1 TO 5
30 FOR j=1 TO 5
40 GO SUB 9000: LET a(i,j)=li
50 NEXT J
60 NEXT i
70 FOR i=1 TO 5
80 FOR j=1 TO 5
90 PRINT a(i,j); " ";
100 NEXT j
110 PRINT
120 NEXT i
130 REM
140 REM dalszy ciąg programu
150 REM
500 STOP
```

Czytane wartości zmiennej tablicowej a(i,j) wykonujemy w programie przez wywołanie podprogramu instrukcją GOSUB 9000 i podstawienie pod zmienną a(i,j) liczby Li.
Należy przy tym zauważyć, że dane pisane w zwykły sposób są także poprawnie czytane. Korzystanie z podprogramu wykazuje, że czytanie danych z powtórzeniami jest bardzo korzystne i przydatne, gdyż w istotny sposób skraca liczbę znaków wprowadzanych z klawiatury.
Dla mikrokomputerów pracujących w systemie CP/M z interpreterem BASIC-80 firmy MICROSOFT podprogram czytania wygląda następująco:

```
9000 IF lp>1 THENLET lp=lp-1: RETURN
9010 INPUT a$: LET mi=INSTR (a$."•")
9020 IF mi=0 THEN LET li=VAL(a$): RETURN
9030 LET lp=VAL (RIGHT$(a$,LEN(a$)-mi))
9040 LET li=VAL(LEFT(a$, mi-1))
9050 RETURN
```

Zmieniając w instrukcji 9010 INPUT a$ na READ a$ można odczytywać dane z instrukcji DATA.

Janusz Szczerbiński

LISTA ROZKAZÓW Z 80

Większość osób piszących programy na swoich komputerach korzysta z języków wysokiego poziomu takich jak Pascal, Basic czy Fortran.

Niekiedy jednak zachodzi konieczność posłużenia się assemblerem lub wręcz kodem maszynowym. Dotyczy to szczególnie procedur blisko związanych ze sprzętem. Programowanie w assemblerze jest dość trudne, choćby z powodu rozbudowanego repertuaru rozkazów procesora.

W tabelce 1 przedstawiliśmy zwięzłą listę operacji realizowanych przez Z80. Zestawienie to będzie idealnym narzędziem dla wszystkich, którzy będą musieli napisać lub zrozumieć krótki program w assemblerze lub kodzie procesora Z80. Napisanie INLINE'a w Turbo Pascalu lub utworzenie niewielkiego RSX'a to dwa typowe zastosowania. Posługiwanie się tą tabelką pozwoli nie wprowadzać „nowych" — nieistniejących rozkazów typu add ix,hl lub podobnych.

Na rys. 2. zestawiono oddzielnie często używane rozkazy przesunięć cyklicznych i arytmetycznych, których zrozumienie powoduje nie rzadko problemy przy analizie programu.

Przedstawione „pomoce naukowe" znajdą zastosowanie dla komputerów z mikroprocesorem Z80: ZX81, ZX Spectrum, Amstrady CPC 464/664//6128, PCW 8256/8512, Meritum, TRS 80, SHARP MZ, Commodore C128 i innych mniej typowych.

Namawiamy użytkowników pozostałych popularnych komputerów, głównie ATARI XL, Commodore C-64, do sporządzenia podobnych zestawień. Profesjonaliści mogliby zająć się INTEL'em 8088 (IBM PC) lub Motorolą 68000 (ATARI ST, AMIGA). Jesteśmy przekonani, że zainwestowanie „paru" godzin w przygotowanie podobnych tabelek dla innych komputerów pozwoli w przyszłości zaoszczędzić wiele czasu, wielu ich użytkownikom. Chętnie je opublikujemy.

Jonasz Mayer

(C) JM 1988 — LISTA ROZKAZOW MIKROPROCESORA Z80

8 - bitowe rozkazy ladowania rejestrow

	A	B	C	D	E	H	L	(HL)	(BC)	(DE)	(nn)	n
LD A, .	7F	78	79	7A	7B	7C	7D	7E	0A	1A	3Axxxx	3Exx
LD B, .	47	40	41	42	43	44	45	46				06xx
LD C, .	4F	48	49	4A	4B	4C	4D	4E				0Exx
LD D, .	57	50	51	52	53	54	55	56				16xx
LD E, .	5F	58	59	5A	5B	5C	5D	5E				1Exx
LD H, .	67	60	61	62	63	64	65	66				26xx
LD L, .	6F	68	69	6A	6B	6C	6D	6E				2Exx
LD (HL), .	77	70	71	72	73	74	75					36xx
LD (BC), .	02											
LD (DE), .	12											
LD (nn), .	32xxxx											

	A	B	C	D	E	H	L
LD . ,(IX+d)	DD7Exx	DD46xx	DD4Exx	DD56xx	DD5Exx	DD66xx	DD6Exx
LD . ,(IY+d)	FD7Exx	FD46xx	FD4Exx	FD56xx	FD5Exx	FD66xx	FD6Exx
LD (IX+d), .	DD77xx	DD70xx	DD71xx	DD72xx	DD73xx	DD74xx	DD75xx
LD (IY+d), .	FD77xx	FD70xx	FD71xx	FD72xx	FD73xx	FD74xx	FD75xx
LD (IX+d),n	DD36xxxx						
LD (IY+d),n	FD36xxxx						

				S Z H P N C
LD I,A	ED47	LD A,I	ED57	‡ ‡ 0 ‡ 0 -
LD R,A	ED4F	LD A,R	ED5F	‡ ‡ 0 ‡ 0 -

8 - bitowe rozkazy arytmetyczne i logiczne

	A	B	C	D	E	H	L	(HL)	n	(IX+d)	(IY+D)	S Z H P N C
ADD .	87	80	81	82	83	84	85	86	C6xx	DD86xx	FD86xx	‡ ‡ ‡ ‡ 0 ‡
ADC .	8F	88	89	8A	8B	8C	8D	8E	CExx	DD8Exx	FD8Exx	‡ ‡ ‡ ‡ 0 ‡
SUB	97	90	91	92	93	94	95	96	D6xx	DD96xx	FD96xx	‡ ‡ ‡ ‡ 1 ‡
SBC	9F	98	99	9A	9B	9C	9D	9E	DExx	DD9Exx	FD9Exx	‡ ‡ ‡ ‡ 1 ‡
AND	A7	A0	A1	A2	A3	A4	A5	A6	E6xx	DDA6xx	FDA6xx	‡ ‡ 1 ‡ 0 0
XOR	AF	A8	A9	AA	AB	AC	AD	AE	EExx	DDAExx	FDAExx	‡ ‡ 1 ‡ 0 0
OR	B7	B0	B1	B2	B3	B4	B5	B6	F6xx	DDB6xx	FDB6xx	‡ ‡ 1 ‡ 0 0
CP	BF	B8	B9	BA	BB	BC	BD	BE	FExx	DDBExx	FDBExx	‡ ‡ ‡ ‡ 1 ‡
INC	3C	04	0C	14	1C	24	2C	34		DD34xx	FD34xx	‡ ‡ ‡ ‡ 0 -
DEC	3D	05	0D	15	1D	25	2D	35		DD35xx	FD35xx	‡ ‡ ‡ ‡ 1 -

		S Z H P N C	
CPL	2F	- - 1 - 1 -	(neguj akumulator)
NEG	ED44	‡ ‡ ‡ ‡ 1 ‡	(zmien znak akumulatora)
DAA	27	‡ ‡ ‡ ‡ - ‡	(korekcja dziesietna)

16 - bitowe rozkazy ladowania rejestrow

	BC	DE	HL	SP	IX	IY
LD .. ,nn	01xxxx	11xxxx	21xxxx	31xxxx	DD21xxxx	FD21xxxx
LD .. ,(nn)	ED4Bxxxx	ED5Bxxxx	2Axxxx	ED7Bxxxx	DD2Axxxx	FD2Axxxx
LD (nn),..	ED43xxxx	ED53xxxx	22xxxx	ED73xxxx	DD22xxxx	FD22xxxx
LD SP			F9		DDF9	FDF9

	BC	DE	HL	AF	IX	IY
PUSH ..	C5	D5	E5	F5	DDE5	FDE5
POP ..	C1	D1	E1	F1	DDE1	FDE1

EX DE,HL	EB		EX (SP),HL	E3
EX AF,AF'	08		EX (SP),IX	DDE3
EXX	D9	(BC-BC',DE-DE',HL-HL')	EX (SP),IY	FDE3

16 - bitowe rozkazy arytmetyczne

	BC	DE	HL	SP	IX	IY	S Z H P N C
INC ..	03	13	23	33	DD23	FD23	- - - - - -
DEC ..	0B	1B	2B	3B	DD2B	FD2B	- - - - - -
ADD HL,..	09	19	29	39			- - ‡ - 0 ‡
ADC HL,..	ED4A	ED5A	ED6A	ED7A			‡ ‡ ‡ ‡ 0 ‡
SBC HL,..	ED42	ED52	ED62	ED72			‡ ‡ ‡ ‡ 1 ‡
ADD IX,..	DD09	DD19		DD39	DD29		- - ‡ - 0 ‡
ADD IY,..	FD09	FD19		FD39		FD29	- - ‡ - 0 ‡

rozkazy przesuniec bloku i szukania

			S Z H P N C
LDI	EDA0	(LD(DE),(HL);INC HL;INC DE;DEC BC)	- - 0 ‡ 0 -
LDIR	EDB0		- - 0 0 0 -
LDD	EDA8	(LD(DE),(HL);DEC HL;DEC DE;DEC BC)	- - 0 ‡ 0 -
LDDR	EDB8		- - 0 0 0 -
CPI	EDA1	(LD A,(HL);INC HL;DEC BC)	‡ ‡ ‡ ‡ 1 -
CPIR	EDB1		‡ ‡ ‡ ‡ 1 -
CPD	EDA9	(LD A,(HL);DEC HL;DEC BC)	‡ ‡ ‡ ‡ 1 -
CPDR	EDB9		‡ ‡ ‡ ‡ 1 -

rozkazy dotyczace pojedynczych bitow

	A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)
BIT 0	CB47	CB40	CB41	CB42	CB43	CB44	CB45	CB46	DDCBxx46	FDCBxx46
BIT 1	CB4F	CB48	CB49	CB4A	CB4B	CB4C	CB4D	CB4E	DDCBxx4E	FDCBxx4E
BIT 2	CB57	CB50	CB51	CB52	CB53	CB54	CB55	CB56	DDCBxx56	FDCBxx56
BIT 3	CB5F	CB58	CB59	CB5A	CB5B	CB5C	CB5D	CB5E	DDCBxx5E	FDCBxx5E
BIT 4	CB67	CB60	CB61	CB62	CB63	CB66	CB65	CB66	DDCBxx66	FDCBxx66
BIT 5	CB6F	CB68	CB69	CB6A	CB6B	CB6C	CB6D	CB6E	DDCBxx6E	FDCBxx6E
BIT 6	CB77	CB70	CB71	CB72	CB73	CB74	CB75	CB76	DDCBxx76	FDCBxx76
BIT 7	CB7F	CB78	CB79	CB7A	CB7B	CB7C	CB7D	CB7E	DDCBxx7E	FDCBxx7E
RES 0	CB87	CB80	CB81	CB82	CB83	CB84	CB85	CB86	DDCBxx86	FDCBxx86
RES 1	CB8F	CB88	CB89	CB8A	CB8B	CB8C	CB8D	CB8E	DDCBxx8E	FDCBxx8E
RES 2	CB97	CB90	CB91	CB92	CB93	CB94	CB95	CB96	DDCBxx96	FDCBxx96
RES 3	CB9F	CB98	CB99	CB9A	CB9B	CB9C	CB9D	CB9E	DDCBxx9E	FDCBxx9E
RES 4	CBA7	CBA0	CBA1	CBA2	CBA3	CBA4	CBA5	CBA6	DDCBxxA6	FDCBxxA6
RES 5	CBAF	CBA8	CBA9	CBAA	CBAB	CBAC	CBAD	CBAE	DDCBxxAE	FDCBxxAE
RES 6	CBB7	CBB0	CBB1	CBB2	CBB3	CBB4	CBB5	CBB6	DDCBxxB6	FDCBxxB6
RES 7	CBBF	CBB8	CBB9	CBBA	CBBB	CBBC	CBBD	CBBE	DDCBxxBE	FDCBxxBE
SET 0	CBC7	CBC0	CBC1	CBC2	CBC3	CBC4	CBC5	CBC6	DDCBxxC6	FDCBxxC6
SET 1	CBCF	CBC8	CBC9	CBCA	CBCB	CBCC	CBCD	CBCE	DDCBxxCE	FDCBxxCE
SET 2	CBD7	CBD0	CBD1	CBD2	CBD3	CBD4	CBD5	CBD6	DDCBxxD6	FDCBxxD6
SET 3	CBDF	CBD8	CBD9	CBDA	CBDB	CBDC	CBDD	CBDE	DDCBxxDE	FDCBxxDE
SET 4	CBE7	CBE0	CBE1	CBE2	CBE3	CBE4	CBE5	CBE6	DDCBxxE6	FDCBxxE6
SET 5	CBEF	CBEB	CBE9	CBEA	CBEB	CBEC	CBED	CBEE	DDCBxxEE	FDCBxxEE
SET 6	CBF7	CBF0	CBF1	CBF2	CBF3	CBF4	CBF5	CBF6	DDCBxxF6	FDCBxxF6
SET 7	CBFF	CBFB	CBF9	CBFA	CBFB	CBFC	CBFD	CBFE	DDCBxxFE	FDCBxxFE

	S Z H P N C
BIT	? ‡ 1 ? 0 -
SET	- - - - - -
RES	- - - - - -

rozkazy wejscia/wyjscia

								S Z H P N C
IN .,(C)	ED78	ED40	ED48	ED50	ED58	ED60	ED68	‡ ‡ 0 ‡ 0 -
OUT (C),.	ED79	ED41	ED49	ED51	ED59	ED61	ED69	- - - - - -
IN A,n	DBxx							- - - - - -
OUT n,A	D3xx							- - - - - -

			S Z H P N C
INI	EDA2	(IN (HL),(C);INC HL;DEC B)	? ‡ ? ? 1 -
INIR	EDB2		? 1 ? ? 1 -
IND	EDAA	(IN (HL),(C);DEC HL;DEC B)	? ‡ ? ? 1 -
INDR	EDBA		? 1 ? ? 1 -
OUTI	EDA3	(OUT (C),(HL);INC HL;DEC B)	? ‡ ? ? 1 -
OTIR	EDB3		? 1 ? ? 1 -
OUTD	EDAB	(OUT (C),(HL);DEC HL;DEC B)	? ‡ ? ? 1 -
OTDR	EDBB		? 1 ? ? 1 -

rozkazy sterujace CPU

			S Z H P N C
NOP	00	(pusta operacja)	- - - - - -
HALT	76		- - - - - -
EI	FB	(zezwol na przerwania)	- - - - - -
DI	F3	(zabron przerwan)	- - - - - -
IM 0	ED46		- - - - - -
IM 1	ED56		- - - - - -
IM 2	ED5E		- - - - - -
CCF	3F	(neguj carry flag)	- - ‡ - 0 ‡
SCF	37	(ustaw carry flag)	- - 0 - - 1

operacje przesuwu cyklicznego i arytmetycznego

	A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)
RR	CB1F	CB18	CB19	CB1A	CB1B	CB1C	CB1D	CB1E	DDCBxx1E	FDCBxx1E
RL	CB17	CB10	CB11	CB12	CB13	CB14	CB15	CB16	DDCBxx16	FDCBxx16
RRC	CB0F	CB08	CB09	CB0A	CB0B	CB0C	CB0D	CB0E	DDCBxx0E	FDCBxx0E
RLC	CB07	CB00	CB01	CB02	CB03	CB04	CB05	CB06	DDCBxx06	FDCBxx06
SRA	CB2F	CB28	CB29	CB2A	CB2B	CB2C	CB2D	CB2E	DDCBxx2E	FDCBxx2E
SLA	CB27	CB20	CB21	CB22	CB23	CB24	CB25	CB26	DDCBxx26	FDCBxx26
SRL	CB3F	CB38	CB39	CB3A	CB3B	CB3C	CB3D	CB3E	DDCBxx3E	FDCBxx3E

	S Z H P N C			S Z H P N C
RR/RL	‡ ‡ 0 ‡ 0 ‡	RRCA	0F	- - 0 - 0 ‡
RRC/RLC	‡ ‡ 0 ‡ 0 ‡	RLCA	07	- - 0 - 0 ‡
SRA/SLA	‡ ‡ 0 ‡ 0 ‡	RRA	1F	- - 0 - 0 ‡
SRL	‡ ‡ 0 ‡ 0 ‡	RLA	17	- - 0 - 0 ‡
		RLD(HL)	ED6F	‡ ‡ 0 ‡ 0 -
		RRD(HL)	ED67	‡ ‡ 0 ‡ 0 -

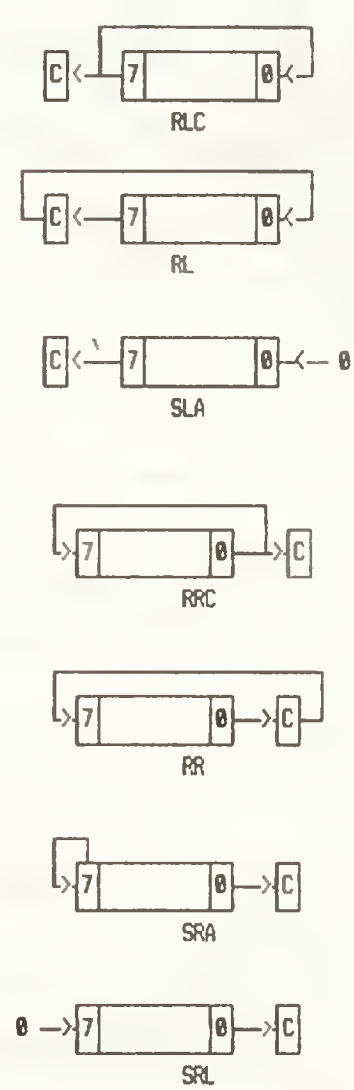
rozkazy sterujace: skoku, procedury, itp.

war.	JP	JR	CALL	RET			RST	
Z	CAxxxx	28xx	CCxxxx	C8	RETI	ED4D	00	C7
NZ	C2xxxx	20xx	C4xxxx	C0	RETN	ED45	08	CF
C	DAxxxx	38xx	DCxxxx	D8	DJNZ	10xx	10	D7
NC	D2xxxx	30xx	D4xxxx	D0			18	DF
PE	EAxxxx		ECxxxx	E8			20	E7
PO	E2xxxx		E4xxxx	E0			28	EF
M	FAxxxx		FCxxxx	F8			30	F7
P	F2xxxx		F4xxxx	F0			38	FF
(bez)	C3xxxx	18xx	CDxxxx	C9				

bit	flaga	nazwa (rejestr F)
7	S	Sign flag
6	Z	Zero flag
5	X	- - -
4	H	Half carry flag
3	X	- - -
2	P/V	Parity/oVerflow flag
1	N	Add/Subtract flag
0	C	Carry flag

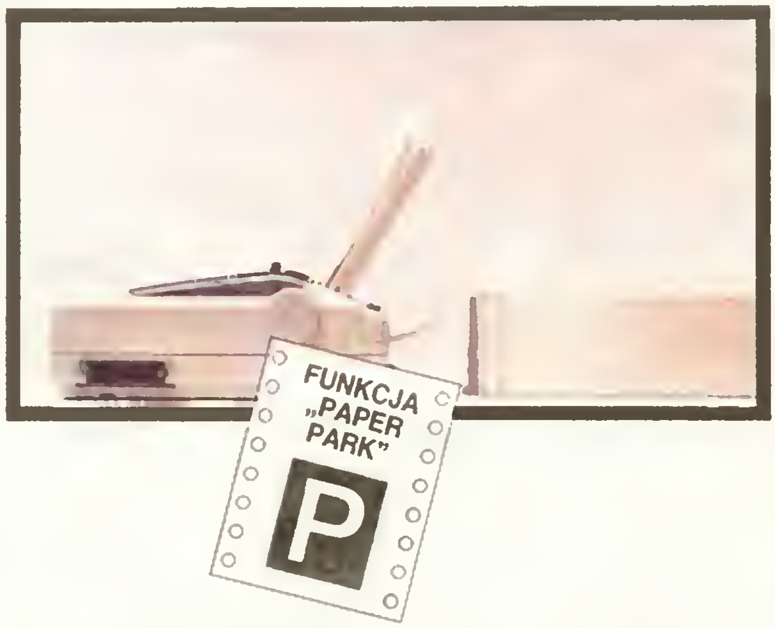
(oznaczenia)
- 1 - ustawiona
- 0 - opuszczona
- ‡ - zalezy od op.
- ? - nieokreslona
- - - niezmieniona

Rozkazy przesunięć cyklicznych i arytmetycznych





A oto nowa gwiazda: star LC-10



Najwyższy poziom technologii japońskiej:	Funkcja „**PAPER PARK**”. możliwość stosowania pojedyńczych stron oraz papieru z perforacją.
Szeroki wybór zestawów znaków:	8 różnych krojów wbudowanych w drukarkę i znaki ASCII/IBM; wersja Commodore C-64/128; znaki dowolnie programowane.
Łatwość użytkowania:	Kilkanaście funkcji wybieranych za pomocą przycisków na obudowie.
Szybkość druku:	120 lub 144 zn/sek w trybie standard; 30 lub 36 zn/sek w trybie korespondencyjnym.
Druk kolorowy:	Wersja LC-10 colour, drukuje w 7 kolorach!
Rewelacyjne ceny:	LC-10 lub LC-10C (do C-64/128) — DM 450 LC-10 colour lub LC-10C colour — DM 590 plus transport: DM 40, kabel: DM 20
Pełna oferta:	Oczywiście oferujemy Państwu pełną gamę drukarek Star łącznie z najnowszą **drukarką laserową LS-08** (8 str/ min), 1MB, kompatybilna z HP Laser Jet II) za DM 4500.

Wyłączny autoryzowany przedstawiciel na Polskę:

ABC Data
peripherals & computer systems

star
Twoja drukarka.

ABC Data Im- und Export GmbH
Augustastraße 40. 5300 Bonn 2, RFN
tel. 0228/35.44.80.-90. telex 88.55.66

ABC Computersystems
Alt Moabit 80
1000 Berlin 21
tel. 391.50.99
Telex 181.365

ABC Data GmbH
Ditmar-Koel-Str. 13
2000 Hamburg 11
tel. 31.40.03
Telex 21.66.002

Interface
do ATARI i do SPECTRUM
drukarki, joysticki, do dowolnego magnetofonu
również — TURBO.
Sterowniki dzwonków szkolnych, makiet, reklam i inne.
PAWTRONIK Warszawa
tel. 659-38-44
D-206

Masz LDW 2000 SUPER?
Potrzebujesz program inicjujący „GAME LOADER".
— 180KB na stronę dyskietki,
— 3 x szybsze czytanie programów,
— nazwy programów do 19 dowolnych znaków
Informacje po nadesłaniu koperty zwrotnej ze znaczkiem
Bartłomiej Lisek, ul. Reymonta 47/4 45-066 Opole
G-131

Agencyjny Zakład Usługowy SPHW
poleca usługi w zakresie:
— serwis mikrokomputerów SPECTRUM, C-64, TIMEX
— gry i programy na SPECTRUM, ATARI, C-64
Warszawa ul. Puławska 102
tel. 44-87-89
czynny 12–19. Gwarancja, rachunki. Zakład przeniesiony z ul. Mokotowskiej 61.
D-197

—ATARI—
ZX SPECTRUM

INSTRUKCJE
OPISY
LITERATURA

Szkoły i Kluby — Zniżka
Katalogi — Gratis
Co piąty program — Gratis
Wysyłka na cały kraj
Wypożyczalnia programów
D.H. „Sezam" II p. g. 16.00—19.00
00-849 Warszawa UPT 66, skr. p. 14.
D-163

PC *plus* Usługi Komputerowe
91-160 Łódź, ul. Mencla 44, tel.557575
KOMPLEKSOWA OFERTA DLA MIKROKOMPUTERA
SpectraVideo SVI-738 :
* Bogate,własne oprogramowanie systemowe, narzędziowe i aplikacyjne.
* Poprawa jakości wyświetlania ekranu.
* Rozszerzenie możliwości graficznych
D-202

„BETA B"
AGENCJA INFORMATYCZNA
Telefon 632-935 690-385
41-200 Sosnowiec, skrytka 254
oferuje również wysyłkowo:
Programy, Instrukcje, Literaturę dla komputerów
ACORN AMSTRAD ATARI COMMODORE SHARP IBM
K-106

ATASERW
43-100 TYCHY
ul. Lencewicza 46/3
tel. 27-69-66
oferuje świetne rozwiązania sprzętowe
do ATARI XL/XE:
1. Ataserw DOS-pierwszy DOS na kartridżu.
2. TOP DRIVE 1050 — wersja 3.0.
3. PIÓRO ŚWIETLNE — wersja 4.0.
4. BASIC XE-kartridż oraz kartridże z dowolnym programem.
5. Rozszerzenia pamięci do 128 i 256 kB.
6. Programator pamięci EPROM.
7. Interfejs Atari — Centronics.
8. Oprogramowanie użytkowe.

Informacje po otrzymaniu koperty zwrotnej, dla instytucji rachunki.
K-212

WOJEWÓDZKIE PRZEDSIĘBIORSTWO HANDLU WEWNĘTRZNEGO ODDZIAŁ W TYCHACH

VIDEOBIT
43-100 Tychy, aleja ZMP 77
tel. 27-69-75
Poleca dla j.g.u.:
- **minikomputery 8-bitowe (Atari, Commodore, Schneider-Amstrad)**
- **minikomputery 16-bitowe kompatybilne z IBM PC**
- **drukarki 10" i 15" firm STAR, EPSON, AMSTRAD**
- **magnetowidy**
- **kamery video**
- **anteny satelitarne**
- **aparaturę badawczo-naukową**

Zapewniamy o atrakcyjnych cenach.
G-7

ATARI ● SPECTRUM ● SHARP ●
nowości programowe
Instalowanie TURBO w ATARI-bezpłatnie!
horoskopy indywidualne. NOWOŚĆ!!!
wysyła pocztą oraz oferuje na miejscu
ASTRO-KOMPUTER STUDIO
54-515 WROCŁAW, Gdacjusza 39 (a-bus 139)
G-133

MICROMAN
Programy na Atari XL/XE. Spectrum 48 KB, Commodore 16/116/+ 4 na miejscu lub za zaliczeniem pocztowym. Informacje za załączeniem koperty i znaczka pocztowego. 40–181 Katowice, ul. Osikowa 66, tel. 58-51-06.
D-204

STUDIO KOMPUTEROWE
ATARI-BAJT
ATARI ● AMSTRAD
COMMODORE ● SPECTRUM
oferuje:
najnowsze programy edukacyjne, użytkowe, gry, opisy
interfejsy do wpisywania programów z każdego magnetofonu i
TURBO — interface ATARI
FINAL II — C 64
tel. 20-80-34 Warszawa do zamówienia katalogi-gratis
D-151

ATARI XL/XE
Bardzo duży wybór oprogramowania z opisami, nowości, co dziesiąta gra gratis, wysyłka na cały kraj, katalog po przesłaniu koperty i znaczka, porady dla początkujących, gwarancja jakości. **TANTAL** Warszawa ul. Staszica 13 (przy DT Wola)
tel. 32-70-60 **Zapraszamy**
(SB 11)

ZX SPECTRUM!
Nowy, oryginalny zestaw programów eksperymentalno-użytkowych TOTO (DL, SL, Ex, ZS)!
typowania komputerowe
wprowadzanie własnych liczb
losowanie graficzne
sprawdzanie (60 kuponów w 5s)
przykładowe wygrane
inne możliwości
Cena programu dla jednej gry — 880 zł + cena kasety.
Zamówienia: MASTER BIT
61-660 Poznań 31
skr. 56
D-182

ATARI
Zakład Elektroniczny „TURBO" oferuje sprawdzone, niezawodne interfejsy, które pozwolą zaoszczędzić Ci wiele dolarów i złotówek.
1) przystawka umożliwiająca współpracę komputera z dowolnym magnetofonem — w pełni zastępuje magnetofon firmowy, a ponadto posiada:
   — wyłącznik auto-stopu
   — układ do kopiowania magnetofon-magnetofon
   — bardzo wysoką czułość odczytu
2) przystawka foniczna z głośnikiem umożliwiająca odbiór fonii bez konieczności przestrajania komputera i telewizora
3) szeroki wybór gier i programów użytkowych na ATARI

Piszcie na adres: Z.E. TURBO
39-300 MIELEC 3 skr. p. 4
tel. 65 wewn. 26
ROCZNA GWARANCJA! RACHUNKI!
G-137

- **7000 typów elementów elektronicznych to nasz program**
- **777 układów scalonych to nasza oferta stała**
- **diody, wyświetlacze, tranzystory, kondensatory, kwarce i rezystory z importu oraz dekodery PAL według potrzeb klienta**
- **najpełniejsza informacja techniczna z oryginalnych katalogów producenta**
- **towar zawsze wysokiej gwarantowanej jakości**
- **stabilne ceny**
- **ewentualna kompletacja pod zamówienie tylko w specjalistycznym sklepie**

Przedsiębiorstwo
Obrotu Maszynami i Surowcami
„BOMIS" PSD nr 10
61-825 Poznań
ul. Krysiewicza 5, Tel. 532-531
(SB 9)

Przedsiębiorstwo Zagraniczne KAREN
ul.Obrońców 23,
03-933 Warszawa
tel. 17 84 10
tlx 813948 kren pl

Szanowny Panie Dyrektorze,

Dziękujemy za zainteresowanie naszą firmą.
Z przyjemnością informujemy, że możemy zaspokoić wszystkie potrzeby Pana Przedsiębiorstwa określone w skierowanym do nas zapytaniu.
1. Oferujemy niezawodne i jednolite systemy komputerowe typu PC/XT/AT/386.
2. Instalujemy adaptery i oprogramowanie sieciowe ETHERNET.
3. Do Zakładu Poligrafii polecamy zestaw ATARI ST DESKTOP PUBLISHING - bogato oprogramowany i oczywiście z polskimi literami.
4. Do Klubu i Szkoły proponujemy ośmiobitowe ATARI XE.
Proszę nie niepokoić się o "wsad dewizowy" - to wszystko jest za złotówki.
Sprzęt objęty jest roczną gwarancją a przy odbiorze będzie mogł Pan uzupełnić swoje zbiory oprogramowania i literatury.

Z poważaniem,

DZIAŁ HANDLOWY

SB — 4

MERA-ELZAB

„Systemy wspomagające umożliwiają nowoczesne projektowanie uruchamianie i testowanie systemów mikroprocesorowych. Zakłady Urządzeń Komputerowych **MERA-Elzab** (41—808 Zabrze ul. Kruczkowskiego 39 tel. 72-20-21) produkują systemy wspomagające RTDS wyposażone w emulatory układowe mikroprocesorów:

- w systemie RTDS-8 m 3
  Intel 8080 i 8085, Zilog 80, Intel 8048
- w systemie RTDS-16 i RTDS/2-PC
  Intel 8088 i 8086

W przygotowaniu emulatory mikroprocesorów: Intel 8031/51, 80186/188, 80286, 80386, V20, V30.
Zakład Systemów Automatyki Kompleksowej PAN (44-100 Gliwice, ul. Bałtycka 5 tel. 31-08-11 wew. 197) zaprasza w każdy ostatni piątek miesiąca o godzinie 11.00 na seminarium połączone z pokazem nt. systemów wspomagających z rodziny RTDS.

(SB 10)

CZĘŚCI ELEKTRONICZNE

pochodzenia zagranicznego za złotówki
— układy TTL, LS, CMOS, inne
— mikroprocesory, pamięci, dyskietki.
Krótkie terminy dostaw, atrakcyjne ceny.
Pośredniczymy także przy sprzedaży.

"MARITEX" tel. 22-02-89
Gdynia, Bat. Chłopskich 3, tlx 054622

(SB 6)

REKLAMUJ SIĘ W BAJTKU

COMPUTER SERVICE

IBM® PC·XT/AT
KOMPATYBILNE

K-57

ZX·Spectrum
Amstrad TIMEX
Schneider Sharp

® Reg Trade Marks of IBM Corporation

PMS ELEKTRONIK
☎ 37·76·65 WARSZAWA
ul. LEGIONOWA 23, ✉ 01-343

UDOSKONALENIA
PROGRAMOWE I SPRZĘTOWE
DLA WSZYSTKICH MODELI ATARI
ORAZ KOMPUTERÓW PRACUJĄCYCH POD SYSTEMEM MS-DOS
WYSYŁA POCZTA
agencja mikro-komputerowa * AMIGO *
41-200 Sosnowiec K-91

ATARI
COMMODORE
C16 C64 C+4
— Największy wybór programów
— Ekspresowa obsługa
— Ceny — prawie za darmo
— Literatura
— Katalogi gratis
KOMBIT
Zwycięstwa 143/6
75-950 KOSZALIN
G-124

JOYSTICK SERVICE
Nie popełniaj błędu kupując nowy joystick! Zlecając nam naprawę zyskujesz wielokrotnie. Twój ulubiony joystick będzie tańszy, oraz trwalszy od nowego
• wymiana standardowych styków na mikroprzełączniki
•• naprawa
Prowadzimy ekspedycję pocztową. Szczegółowe informacje po nadesłaniu koperty zwrotnej.
Zgłoszenia: Studio komputerowe SEZAM D.H. "SEZAM" II p. — czwartki 16⁰⁰ — 19⁰⁰
Korespondencja: JOYSTICK SERVICE 02-770 Warszawa 130 skr. poczt. 102 tel. grzeczn. 41-22-22
D-183

„ATARES" spółka z o.o. Chorzów Batory, Jesionowa 3, tel. 417-573 oferuje: oprogramowanie „Atari", „Commodore", „Spectrum", system „Blizzard Turbo", magnetofon „Atari" po przeróbce czyta 10 razy szybciej i pewniej, roczna gwarancja, cena 33.200 złotych system „Crystal Sound" do digitalizacji dźwięku „Atari", cena 15.500,- komis, skup, sprzedaż komputerów, RTV, video.
K-217

„MIKROELEKTRONIKA OD PODSTAW DLA KAŻDEGO"
Błyskawicznie, tanio, rewelacyjną metodą — od prawa Ohma do poznania możliwości i wnętrza mikrokomputerów. Wysyłkowa sprzedaż wiedzy oraz płytek do samodzielnego montażu mikrokomputera CA80 ukierunkowanego na sterowania.
Szczegółowa wielotomowa dokumentacja. Informacje — koperta zwrotna ze znaczkiem „MIK" Stanisław Gardynik 05-090 Raszyn
D-157

Dyskietki firmowe PRECISION, BASF, 3M, MAXELL, Bonus, Dysan, Verbatim — najtaniej od dostawcy z U.S.A.
Informacje, cenniki: „Elektronika", Kraków, Proszowicka 9, tel. 34-19-10
D-185

STAR SG — 10 Commodore zamienię na Centronikc lub sprzedam
tel. 6627-237
D-194

Układy scalone COMMODORE poleca
Uniservice skrytka 40
33-110 Tarnów 2
G-138

Bogaty wybór polskich wersji najciekawszych gier na ZX SPECTRUM oferują:
Jarosław Stępień, Chojnowska 13 m 10 59-220 Legnica oraz
Jarosław Szarsio, Kilińskiego 27 m 17 96-300 Żyrardów
G-141

Najmłodszy miliarder

Wiliam Gates, dla Amerykanów po prostu: Bill — cudowne dziecko (32 lata) biznesu, którego droga do sukcesu potwierdza głęboko zakorzenioną w społeczeństwie legendę pucybuta. Swoje miliony Bill zarobił sam.

W ubiegłym roku fachowcy z tygodnika „US News and World Report" ocenili go na 750 mln, co dawało mu wówczas 16 miejsce na liście stu najbogatszych Amerykanów. W tym roku Wiliam H. Gates III — tak brzmi jego pełne nazwisko, dostał się już do klubu miliarderów wykreowanego przez FORTUNE — pismo, którego 48 reporterów i korespondentów zagranicznych oraz setki konsultantów i analityków pisma przetrząsnęło świat cały w poszukiwaniu miliardowych portfeli.

W rezultacie powstał — zdaniem autorów — najbardziej zwięzły i wiarygodny ranking najbogatszych, w których gronie najmłodszym jest właśnie Bill z kapitałem prawie dwa razy większym niż w roku ubiegłym (1,4 mld). Na tej liście daje mu to zaledwie 79 pozycję, a do prowadzących stawkę sułtana Brunei czy amerykańskiej rodziny Marsów brakuje mu — kolejno — dwadzieścia i dziewięć razy więcej. Zważywszy jednak dynamikę wzrostu jego dochodów łatwo wyliczyć można, kiedy młody jeszcze Gates osiągnie czoło stawki miliarderów — po amerykańsku to lepiej: bilionerów.

Póki co zadje się nie zwracać uwagi na wysokość konta zadawalając się tym, co sprawia mu największą przyjemność — tworzeniu oprogramowania. Jego udział w Microsoft Corp. której jest prezesem, głównym strategiem, sympatycznym szefem i najlepszym sprzedawcą wynosi prawie 42 procent.

Pozwala mu to na niepodzielne panowanie nad spółką założoną 12 lat temu wspólnie z przyjacielem — Paulem Allenem, który tak jak i on wyrzucony został wcześniej z Harwardu.

Nie od razu jednak Microsoft zbudowano. Paul i Bill zaczynali w sposób dla tej branży typowy: w garażu. Po kilku udanych próbach sprzedaży programów udało im się zainteresować swoimi produktami potężny IBM. W 1981 roku koncern ten zakupił dla 11 milionów wyprodukowanych przez siebie mikrokomputerów FC system operacyjny opracowany przez Billa. I zaczęło się: wartość sprzedawanego przez Microsoft Corp. oprogramowania skoczyła z 32 mln dolarów w 1982 roku do 140 w roku 1986 i 750 w roku ubiegłym. Przyszłość spółki jest zapewniona: szefowie IBM zapewnili, że ich najnowsza generacja komputerów osobistych sterowana będzie również wg systemu oferowanego przez Microsoft.

Ale Billowi to już nie wystarcza — w szybkim tempie rozrasta się dział wydawniczy spółki, oferując tytuły książek niekoniecznie z dziedziny „komputoroznawstwa".

Wszyscy śledzący karierę najmłodszego w USA (i na świecie) miliardera podkreślają, iż Bill Gates okazał się nie tylko komputerowym geniuszem, lecz także zręcznym biznesmenem. W odróżnieniu od swych rówieśników z „innych garaży" od razu postawił na profesjonalizm w zarządzaniu, przyjmując na stanowisko dyrektora doświadczonego Jona Shirleya (50 lat). Dla siebie pozostawił strategię, taktykę i technologię. Brzmi to poważnie i Bill zajmuje się tymi sprawami z impetem, pasją i przyjemnością. Bawi go to co robi i — jak twierdzą współpracownicy — zdarza się, że ważnym klientom sam demonstruje zalety swoich programów.

Jako naczelny strateg i właściciel firmy panuje nad nią doskonale:

— Każda rozmowa z nim o interesach jest wyczerpująca — twierdzi Jon Shirley. — Dedukuje błyskawicznie szukając luk w twoich argumentach. Niech Bóg ma w opiece tego, kto rozpocznie z Billem fachowy dyskurs nie przygotowując się doń solidnie w domu.

A Bill jest przygotowany zawsze. W biurze pojawia się o 9.30 przyjeżdżając swoim czterodrzwiowym Jaguarem; kończy prawie zwykle ok. 1 w nocy — i tak przez sześć dni w tygodniu. Często zdarza się, że poprzez swój domowy komputer wydaje dyspozycje personelowi nad ranem i w niedzielę.

Ile zarobi w roku przyszłym?

Dotychczasowa dynamika rozwoju wskazuje, że około 3 mld dolarów, co przesunęło by go to znacznie do przodu zarówno na liście amerykańskiej (w okolice drugiego miejsca), jak i w klubie miliarderów.

Ale Bill nie powiedział jeszcze ostatniego słowa.

Franciszek Penczek

HEJ KOLĘDA, KOLĘDA...

Zbliżają się święta, a z nimi choinki, prezenty i kolędy. Proponuję stworzyć świąteczny nastrój przy pomocy naszego Spectrum. Nauczymy je w tym celu grać kilka najpopularniejszych kolęd. Wystarczy wprowadzić i uruchomić któryś z zamieszczonych programów. Wesołych Świąt!

Marcin Borkowski

```
0 REM Dzisiaj w Betlejem
10 RESTORE 200: GO SUB 100
20 RESTORE 200: GO SUB 100
30 RESTORE 240: GO SUB 100
40 RESTORE 240: GO SUB 100
50 RESTORE 270: GO SUB 100
60 RESTORE 270: GO SUB 100
70 RESTORE 290: GO SUB 100
90 STOP
100 READ ilenut
110 FOR i=1 TO ilenut
120 READ czas,nuta
130 BEEP czas/3,nuta
140 NEXT i
150 RETURN
200 DATA 16,2,5,1,5,1,0,1,5
210 DATA 1,7,2,9,1,9,1,7,1,9
220 DATA 1,10,1,12,1,14,2,12
230 DATA 2,10,2,9,4,7
240 DATA 10,2,12,1,12,1,10
250 DATA 1,9,1,7,2,5,1,5
260 DATA 1,0,1,5,1,9
270 DATA 6,1,12,1,14,1,12
280 DATA 1,10,1,9,1,10
290 DATA 8,1,12,1,12,2,14
300 DATA 2,12,1,10,1,9,2,7,2,5
```

```
0 REM Przybieżeli do Betlejem
10 RESTORE 200: GO SUB 100
20 RESTORE 200: GO SUB 100
30 RESTORE 230: GO SUB 100
40 RESTORE 230: GO SUB 100
50 STOP
100 READ ilenut
110 FOR i=1 TO ilenut
120 READ czas,nuta
130 BEEP czas/3,nuta
140 NEXT i
150 RETURN
200 DATA 11,1,1,1,0,1,1
210 DATA 1,3,1,5,1,3,1,5
220 DATA 1,6,2,8,2,10,4,8
230 DATA 20,2,13,1.5,8,0.5,8
240 DATA 1,10,1,8,1,6,1,5
250 DATA 2,6,1.5,6,0.5,10
260 DATA 1,8,1,6,1,5,1,3,2,5
270 DATA 2,6,4,8,2,5,2,3,4,1
```

```
0 REM Wśród nocnej ciszy
10 RESTORE 200: GO SUB 100
20 RESTORE 200: GO SUB 100
30 RESTORE 230: GO SUB 100
40 RESTORE 230: GO SUB 100
50 RESTORE 250: GO SUB 100
60 STOP
100 READ ilenut
110 FOR i=1 TO ilenut
120 READ czas,nuta
130 BEEP czas/3,nuta
140 NEXT i
150 RETURN
200 DATA 10,2,5,1,7,1,4,2,5
210 DATA 2,0,1,9,1,9,1,10
220 DATA 1,7,4,9
230 DATA 8,1,5,1,9,1,5,1,9
240 DATA 1.5,10,0.5,7,1,4,1,0
250 DATA 5,1,5,1,5,1,7,1,7,2,5
```

	GIEŁDA „BAJTKA" (tys.zł)	PEWEX BALTONA (USD)	RFN (śred.) (DM)
SINCLAIR			
ZX-Spectrum 48 KB	210	115	90
ZX-Spectrum plus	280	—	99
TIMEX 2048	300	146	—
ZX Spectrum + 2	—	—	150
ZX Spectrum + 3	—	—	280
COMMODORE			
Commodore 64	550	219	299
Commodore 128	800	299	399
Magnetofon 1531	80	48	30
Stacja dyskietek 1541	490	—	355
Stacja dyskietek 1571	550	299	465
C-128 D	1,0 mln	—	888
Drukarka MPS 801	490	—	—
AMIGA 500	2,1 mln	—	999
C-16	190	—	90
C-116	160	—	75
C-+4	210	—	150
ATARI			
ATARI 65 XE	350	125	100
ATARI 130 XE	—	199	230
Stacja dyskietek 1050	380	185	300
LDW 2000	450	199	—
XC-12	90	48	49
AMSTRAD			
464 mon. monitor	700	—	350
464 kolor mon.	800	—	680
6128 mono. monitor	1,0 mln	—	689
6128 kolor monitor	1,2 mln	—	988
Stacja dyskietek do 464	450	—	389
PC 1512 SD MD	3,0 mln	—	998
Dyskietki 3"	7	—	4-9
Dyskietki 5.25"	1.2	—	0.65
Dyskietki 3.5"	6.5	—	4-5 5

Sklep BAJTKA w Bytomiu
ul. Koniewa 6 tel. 81-57-01

ZX-Spectrum 48 K	130.000
ZX-Spectrum plus	175.000
ZX-Spectrum + 2	280.000
SEIKOSHA GP-50S	120.000
Commodore 64	240–270.000
Commodore 128	380.000
Commodore 128 D	—
AMIGA 500	—
Commodore 16	120.000
Commodore 116	100.000
Commodore +4	165.000
Magnetofon 1530	50–60.000
Stacja dysków 1541	270.000
ATARI 65 XE	190.000
ATARI 130 XE	260–290.000
XC-12	60–65.000
Stacja dysków 1050	240–270.000
LDW 2000 Super	270.000
464 mono	340.000

INDYWIDUALNY BANK DANYCH

Aleksander Maciszek, uczeń VII klasy. Posiada Atari 65 XE, stację dysków LDW Super 2000 oraz syntezator KORG, który pragnie wykorzystać do pracy z komputerem. Oprogramowanie: ponad 100 gier, programy muzyczne. Adres: 59-700 Bolesławiec Śl., ul. Zwycięstwa 8/1.

Marcin Sokołowski, lat 13. Posiada mikrokomputer Commodore Plus/4, magnetofon 1531, joystick. Oprogramowanie: gry oraz programy użytkowe. Zainteresowania: elektronika oraz mikrokomputery. Pragnie nawiązać kontakt z „plusowcami" w celu wymiany doświadczeń oraz oprogramowania. Adres: 25-729 Kielce, ul. Urzędnicza 9 A/44.

Grzegorz Jankowski, uczeń lat 14. Posiada mikrokomputer ASI-009, dwa napędy dyskowe 3,5", monitor OMT, drukarkę firmy Enterprise, karty CGA, Hercules. Pragnie nawiązać kontakt w celu wymiany doświadczeń oraz oprogramowania. Adres: 05-805 Otrębusy, ul. Podleśna 15.

Adrian Pelka, uczeń lat 16. Posiada Commodore C-64, magnetofon, dwa joysticki i około 600 programów i gier. Zainteresowania: informatyka, technika video. Proponuje wymianę programów. Adres: 41-907 Bytom, ul. A. Piątka 25a/6.

Tomasz Mieszczak, uczeń 16 lat. Posiada mikrokomputer Atari 800 XL ze stacją dysków LDW 2000. Proponuje wymianę oprogramowania, literatury i opisów programów. Adres: 61-626 Poznań, ul. Winogrady 129.

Wszystkich posiadaczy japońskiego mikrokomputera „M5", firmy SORD COMPUTER Co. prosi o kontakt **Edward Mich** zamieszkały 80-262 Gdańsk 6, ul. Dekerta 3/3

Jacek Kasperczyk, uczeń LO, lat 18. Posiada CPC 6128. Oczekuje wymiany oprogramowania i doświadczeń z innymi posiadaczami tego mikrokomputera. Adres: 28-400 Pińczów, ul. Bat. Chłopskich 119.

Piotr Wojciechowski, inżynier 37 lat. Mikrokomputer Atari 130 XE + stacja dysków. Posiada własne programy zawodowe jak sieci cieplne, straty ciepła, wymienniki ciepła, wentylacja i inne. Całość ułożona w mini poradnik dla inżyniera instalatora. Nawiąże współpracę, wymianę programów. Adres: 02-319 Warszawa, ul. Kopińska 34a/12.

Mirosław Wałasik prosi o kontakt osoby posiadające mikrokomputer TI 99/4A. Adres: 34-400 Nowy Targ, Al. 1000-lecia 30/10.

Marek Kalicki, lat 11. Posiada mikrokomputer w systemie MSX-SONY HIT-BIT 64 KB, magnetofon i joystick oraz oprogramowanie. Adres: 43-300 Bielsko-Biała, ul. Broniewskiego 4/118.

Marlena Cichocka, lat 12. Posiada Commodore 64 oraz około 700 programów, w tym 100 użytkowych. Proponuje wymianę oprogramowania. Adres: 82-500 Kwidzyn, ul. Findera 8/37.

Tomasz Gabarysiak, lat 13. Komputer Commodore C-64, monitor Neptun, magnetofon MTR oraz dwa joysticki. Oprogramowanie: programy muzyczne, użytkowe oraz sporo gier. Zainteresowania: informatyka i sport. Proponuje wymianę literatury oraz doświadczeń. Adres: 66-400 Gorzów Wlkp., ul. Pułaskiego 3a/11.

Violetta Koch, uczennica LO — lat 16. Posiada Atari 800 XL magnetofon XC 12 oraz stację dysków LDW Super 2000. Oprogramowanie: około 500 gier i programów użytkowych. Proponuje wymianę programów z użytkownikami Atari. Adres: 61-673 Poznań, Oś. Kraju Rad 7 „H"/72.

Zbigniew Przeorski, pracownik umysłowy. Posiada nietypowy komputer CASIO FP 1000 pracujący pod systemem CP/M 2.2. Prosi o pomoc w zdobyciu programów użytkowych, głównie bazy danych i arkuszy kalkulacyjnych. Nawiąże kontakt z posiadaczami tego komputera. Adres: Warszawa, ul. Kazury 13 m 3.

JOYSTICK — BŁAHA WAŻNA RZECZ

(mini test użytkowników)

Czytelnicy Bajtka z pewnością kupują go obok magentofonu lub stacji dysków, jako drugie urządzenie peryferyjne.

Wybór joysticka, chociaż nie należy do najpoważniejszych decyzji finansowych, nie jest zupełnie dowolny. Fałszywy oznacza konieczność ponownego zakupu, często już po miesiącu — dwóch.

Redakcja nasza otrzymała przed pół rokiem od pana Tadeusza Trojaka z Łodzi kilka sztuk produkowanych przez niego joysticków MATT. Cztery z nich przeszły generalną „próbę ognia" u naszych kolegów przygotowujących książkę o grach komputerowych, a także w samej redakcji.

Pierwsze opinie wszystkich użytkowników były zgodne — MATT na tle innych konstrukcji krajowych prezentuje się bardzo dobrze. Wyposażony jest m.in. w elegancko wykonaną wtyczkę pasującą do najpopularniejszych w kraju mikrokomputerów — Atari, Commodore, Amstrad, itp. Znakomicie spełniają swoją rolę umieszczone na spodzie joysticka przyssawki. Nawet podczas tak trudnych prób, jak pokonywanie kolejnych etapów „Commando", MATT nie wymaga podtrzymywania drugą ręką, czego nie da się powiedzieć np. o równolegle używanych w redakcji angielskich joystickach.

Opinie po kilku tygodniach użytkowania MATT-ów były również dość zgodne. Wszystkie egzemplarze pracowały bez zarzutu. Narzekali jednak ich użytkownicy na dość oporną pracę joysticka — po pół godzinie gry, szczególnie w różne „zręcznościówki" ręka zaczyna już porządnie boleć. Przydałby się również przycisk „autofire" — wzdychali wielbiciele pozycji typu „kill all".

Sześć miesięcy egzaminu MATT-y zdały na „dobrze". Pierwszy z nich odmówił posłuszeństwa dopiero na tydzień przed zakończeniem próby — podczas wystawy poddał się przy grupowym korzystaniu z „Winter Games". Pozostałe służą nam dalej i dotychczasowi ich użytkownicy nie zmienili ich na zachodnie.

Podsumowując, MATT na naszym pustym rynku złotówkowym (gdzie nie wspominając) jest interesującą propozycją dla użytkownika komputera domowego. Chociaż bardziej polecałbym go tym, którzy używają joysticka w programach użytkowych, np. graficznych, również i fani gier mają chyba z niego pociechę, chociażby ze względu na trwałość konstrukcji. Producent, zresztą, zapowiada liczne udoskonalenia — wprowadzenie mikroswitch'y, wyposażenie MATT-a w „autofire". Te zapowiedzi cieszą. Oby tylko nie wzrosła zbytnio cena. Przy dzisiejszym czarnorynkowym kursie dolara właśnie cena, przynajmniej w moim przypadku, zadecydowałaby o tym, by zamiast do „Pewexu" wybrać się do CSH, gdzie sprzedawane są MATT-y.

Marcin Lutowski



Z PALCEM NA BANK (DANYCH)

O występkach komputerowych hackersów włamujących się za pośrednictwem swej własnej klawiatury, modemu i telefonu do sieci informatycznej wielkich korporacji, ministerstw, czy systemów obronnych napisano już całe tomy.

W Stanach Zjednoczonych i Republice Federalnej Niemiec grasują podobno wyspecjalizowane gangi elektronicznych przestępców, którzy na zlecenie „poprawiają" bankowe konta, kradną zakodowane w sztucznej pamięci maszyn bezcenne dane, czy wszczepiają do sieci wskazanej firmy programowego wirusa niszczącego stopniowo zmagazynowane zasoby informacji. W Nowym Jorku, Sztokholmie, Berlinie Zachodnim i Hanowerze powstały specjalnie wydzielone oddziały policyjne do zwalczania gangsterstwa komputerowego i informatycznego chuligaństwa.

A wszystko zaczyna się i kończy na prostej operacji wejścia do pamięci obcego komputera. Po uzyskaniu telefonicznego połączenia na ekranie monitora elektronicznego poszukiwacza przygód pojawia się prośba systemu, do którego właśnie puka, o podanie kodu. Jeśli słowo — klucz, lub układ liczbowo-literowy jest zgodny z tym, co zaprogramowano w matrycy bezpieczeństwa urządzenia, nie broni ono dostępu do swego wnętrza. Tak jak najlepszy sejf, którego szyfrową kombinację złamali włamywacze

Jeszcze kilka lat temu za Atlantykiem rwano włosy z głowy, wszak specjalnie projektowane ogromne sieci informatyczne za miliony dolarów nie mogły być bezpieczne. Skoro nawet dobrze wyszkoleni amatorzy po miesiącach zabaw w kotka i myszkę, metodą prób i błędów, mogli dobrać się do ich wnętrza. A przecież każdy bank danych nie jest czymś, co zakopuje się w ziemi na lata. Musi działać, dla osób uprawnionych do posługiwania się nim powinien być otwarty w każdej chwili. Nie było więc sensu tworzenie superkodów, skomplikowanych pułapek szyfrowych, skoro jedną z cech chronionego wnętrza powinna być całkowita funkcjonalność.

Na razie poradzono sobie sporządzając w pamięci bronionego komputera dokładną listę osób uprawnionych do korzystania z przechowywanych w nim informacji wraz z ich numerami telefonów. Jeśli od urządzenia żądano informacji podając swoje dane i kod wejściowy, oddzwoniało ono samo do proszącej o wstęp osoby, sprawdzając nazwisko, numer telefonu i hasło. Ten wariant ochronny zwany za Atalntykiem Callback Security System ma też swoje oczywiste wady, ale przede wszystkim jest dość kosztowny, a oprócz tego każda zmiana personelu wymaga odpowiednich przeróbek programu. W sumie amerykańscy eksperci oceniają go niezbyt wysoko, jako „zapchaj dziurę" w dramatycznej sytuacji

Ale wszystko zmieni się prawdopodobnie jeszcze w najbliższych miesiącach. Aż trzy specjalistyczne firmy w Stanach Zjednoczonych w jednakowy sposób podeszły do zagadnienia podsuwając zaniepokojonym szefom korporacji, generałom i bankowcom proste i, jak się je reklamuje, zupełnie niezawodne rozwiązania.

Wszystko zaczęło się od arcyskomplikowanych zamków elektronicznych, broniących dostępu do najściślej strzeżonych w Ameryce pomieszczeń, skonstruowanych jednak tak, aby ci upoważnieni nie mieli specjalnych kłopotów z wejściem. Jak podaje Richard Ernsberger — reporter „Newsweeka" — takie urządzenia działają już w USA i w innych wysokouprzemysłowionych krajach od lat. Prawdopodobnie właśnie w taki sposób wchodzi się do niezwykłej wagi pomieszczenia na 7 piętrze Departamentu Stanu w Waszyngtonie, gdzie zainstalowano mechanizmy „ gorącej linii", łączącej amerykańską stolicę z analogicznym centrum uniknięcia ryzyka przypadkowej wojny jądrowej w Moskwie. Tylko kilka osób może przejść przez dwoje pancernych drzwi z przemyślną śluzą. Należy przypuszczać, że tajemny sezam otwiera się tylko wtedy, gdy do specjalnego czujnika wędruje tych kilka powołanych palców wskazujących.

Jedną z biometrycznych cech zupełnie indywidualnych i niepowtarzalnych dla każdego człowieka są odciski jego palców. Linie papilarne są tym jedynym kluczem otwierającym najdoskonalsze amerykańskie zamki. Ale cały układ drzwiowy do tej pory był zbyt wielki i ciężki, aby mógł służyć również w technice komputerowej. Nie mówiąc już o innych systemach działających na zasadzie porównywania kształtu dłoni z zapamiętaną matrycą.

W tym świetle nowości trzech producentów systemów bezpieczeństwa — firmy Identix z Palo Alto w Kalifornii, przedsiębiorstwa ThumbScan z Oakbrook Terrace w stanie Illinois i spółki Fingermatix z North White Plains w Nowym Jorku, mają wielkie szanse dokonania przewrotu na dobre odizolowującego hackersów od łakomych dla nich kąsków.

Randy Powter — szef i założyciel Identixu poleca czujnik o nazwie Touch Safe nie większy od... typowej „myszy" przyłączonej do komputerowej klawiatury. To właśnie jest ów zamek zabezpieczający wnętrze dowolnego banku danych przed włamaniem. Aby dostać się do informacji z zastrzeżonego systemu wystarczy odcisnąć jeden z naszych palców (niekoniecznie wskazujący) na maleńskiej szybce sensora Touch Safe. Jeśli odcisk tego właśnie palca, należącego do tego właśnie człowieka został wstępnie zapamiętany przez czujnik, jego właściciel ma wolną drogę, system zabezpieczający zostanie odblokowany. Jeśli porównanie matematycznego modelu — matrycy odcisku osoby uprawnionej z liniami odbitymi na szybce (dane z szybki rozkładane są na 250 tysięcy informacji) wypadnie negatywnie, natychmiast uruchamiany jest alarm dla całego układu. Test prawdy trwa dwie sekundy. Touch Safe oferowany jest za 1800 dolarów

W podobny sposób przedstawia czujnik firmy ThumbScan jej rzecznik — Bob Glowienke. Za ten produkt nabywcy będą musieli zapłacić jednak tylko 1000 dolarów. Zaś inteligentny klucz palcowy wytwórni nowojorskiej opatrzony długą nazwą Ridge Reader Mint 11 kosztuje 1500 dolarów.

Producenci zastrzegają jednak, iż ceny dotyczą jedynie samych czujników, do których należy zamówić również programy sprzęgające ich działanie z siecią chronionego systemu.

Randy Powter liczy na poważne i liczne kontrakty. Tego samego spodziewają się przedstawiciele dwóch innych firm. A w najbliższej przyszłości obiecują coś jeszcze doskonalszego — czujniki głosowe, których nie oszuka nawet precyzyjny magnetofon. No cóż, nam wypada jedynie kibicować.

Wojciech Łuczak

KLUB NAD RENEM



W dniach 23.07—30.07 przebywaliśmy na obozie harcerskim w Bremen, w RFN. Gościliśmy u Pfadfinderów (odpowiednik harcerzy). Odwiedziliśmy wtedy Pfadfinder Computer Club, mający swoją siedzibę przy zakładach energetycznych UNH. Pragnąc dowiedzieć się czegoś na temat działalności tego klubu (będącego odpowiednikiem klubu, którego członkami jesteśmy) poprosiliśmy o rozmowę Rainera Nalazka, szefa Pfadfinderów w landzie Bremen.

— Jak powstał klub?

— Idea powstania klubu pochodzi z Polski. Latem 1987 roku przebywaliśmy m.in. na obozie w Perkozie, na Mazurach. Zobaczyliśmy tam działający wasz klub (HarcBajt — dop. J.Ł.). Widzieliśmy również jaką popularnością cieszy się działalność HarcBajtu w Domu Harcerza w Gdansku. Po powrocie do kraju stwierdziliśmy: „Dlaczego oni mają, a my nie?"

Na początek należało postarać się o sprzęt i pomieszczenie dla klubu. Pfadfinderzy są organizacją całkowicie społeczną, czerpiącą finanse ze składek członkowskich, wspomaganą niekiedy przez senatora ds. młodzieży. Zaczęliśmy poszukiwać sponsora. Jeden z naszych przyjaciół pracował w firmie SIEMES. Był zdziwiony ideą założenia klubu komputerowego Pfadfinderów, ale poparł ją. Dzięki niemu, po wielu perypetiach, otrzymaliśmy z SIEMENSa 4 zestawy mikrokomputerów PC-SIEMENS z oprogramowaniem (GEM, BASIC, program graficzny, kalkulacyjny, edytor testu, baza danych, programy narzędziowe). Zaczęliśmy wówczas szukać odpowiedniego pomieszczenia. Po obejrzeniu harcówek stwierdziliśmy, że żadna nie nadaje się na lokal klubowy. Brak było odpowiednich zabezpieczeń i zaplecza. W czasie rozmowy z kolegą z UNH okazało się, że na terenie zakładu znajduje się wolne pomieszczenie znakomicie nadające się na klub komputerowy. W sierpniu 1987 roku wprowadziliśmy się tam.

Nie płacimy tutaj za energię elektryczną ani żadnego czynszu. UNH pomaga ponadto dając papier do drukarek itp.

— Kto prowadzi zajęcia w klubie?

— Od sierpnia 1987 roku rozpoczęliśmy szkolenie instruktorów. Była to grupa starszych Pfadfinderów (20—24 lata). W UNH, gdzie również pracuję, jestem kierownikiem. Z 50 elektroników, 47 zgłosiło chęć społecznej pracy w klubie.

— Jak wyglądał nabór członków? Czy mogą nimi być tylko Pfadfinderzy?

— Członkami klubu mogą być wszyscy chętni. Tyle tylko, że Pfadfinderzy nie płacą za kursy, a inni za 8 tygodniowy kurs płacą 18 marek.

Po 2 stycznia 1988 roku, kiedy to nastąpiło uroczyste otwarcie klubu z udziałem przedstawicieli SIEMENSa, UNH oraz senatora ds. młodzieży, zamieściliśmy w prasie ogłoszenie. Zgłosiło się 237 chętnych. Niestety, nie mogliśmy tylu przyjąć. Przyjęliśmy 48 osób. Pracują one od poniedziałku do piątku w 8-io osobowych grupach pod kierownictwem naszych instruktorów. Szkolenie dotyczy ogólnej obsługi komputerów, prawidłowego posługiwania się oprzyrządowaniem (drukarką, ploterem, dyskietkami). Uczestnicy kursów zwiedzili centrum komputerowe w UNH, zapoznając się z praktycznym wykorzystaniem komputerów. Po zakończeniu kursu uczestnicy dostają dyplomy. Jedynym warunkiem ich otrzymania jest uczestniczenie w co najmniej połowie zajęć klubu.

Wielu „cywilnych" uczestników naszych kursów wstąpiło potem do organizacji Pfadfinderów.

Po wakacjach, jesienią, pojedziemy do centrum komputerowego SIEMENSa. Chcemy również być obecni na CeBicie w Hanowerze, aby móc zapoznać się z najnowszymi osiągnięciami technicznymi w dziedzinie produkcji i oprogramowania komputerów. Na CeBicie istnieje tzw. sala walki komputerowej, gdzie różne firmy i kluby prezentują to, co mają najlepszego. Właśnie tam chcemy się pokazać.

— Czy nawiązaliście już może kontakty z innymi klubami?

— Jeszcze nie. Jesteśmy dopiero w stadium organizacji. Chcielibyśmy uporządkować na razie własne podwórko. Prawdopodobnie dostaniemy jeszcze cztery nowe komputery SIEMENSa. W tej chwili prowadzone są rozmowy z SIEMENSem. Liczymy na ich pomyślny efekt. Mamy w tej chwili trzech udziałowców. Są to SIEMENS, UNH, no i my. Liczymy na pomoc senatu miejskiego. Wspólnymi siłami chcemy doprowadzić do tego, aby jak najwięcej młodzieży w jak najlepszych warunkach korzystało z możliwości klubu.

Na przyszłość: mamy na oku pewien klub komputerowy w Berlinie Zachodnim oraz klub w Hamburgu, działający pod egidą DJH (schroniska młodzieżowe — dop. J.Ł.). Nasz klub odwiedzają grupy podobne do waszej, goszczące u Pfadfinderów. Były tu już dwie grupy z Polski. Głównym celem jest co prawda zawsze zwiedzanie zakładów UNH, lecz gdy powiemy, że przy UNH działa klub komputerowy, wszyscy chcą go zobaczyć. Pod koniec sierpnia oczekujemy wizyty grupy z Francji, a we wrześniu grup z Rygi i z Rostoku.

— Czy możesz mi powiedzieć, jak wygląda nauka informatyki w waszych szkołach?

— Owszem. W szkołach ogólnych nauka informatyki zaczyna się w dziesiątej klasie. Prowadzi je specjalnie szkolony pracownik, tzw. asystent informatyki. Młodzież uczy się języków programowania, np. PASCAL-a

— Czy wiesz coś na temat komputeryzacji w Polsce?

— W naszej telewizji po ostatnim Balticomie (tak, tak, tym w Gdańsku — dop. J.Ł.) dużo mówiło się o komputerach w Polsce. Przeraził mnie szczególnie fakt bezkarnego kopiowania i sprzedawania pirackich kopii programów. Bardzo na to wszyscy zwracają uwagę, gdyż takie praktyki to po prostu kradzież, nieuszanowanie cudzej pracy. U nas też co prawda są piraci, są oni na całym świecie, lecz nie działają tak otwarcie, mają przeciwko sobie prawo i za swoją działalność są surowo karani. Nie rozumiem, dlaczego w Polsce te sprawy nie są ujęte w odpowiednie przepisy.

Nie podobało mi się również częste wykorzystanie komputerów do gier i to najczęściej typu „bij, zabij". Komputer może być wykorzystany do złych celów, np. do gier pornograficznych, równie niebezpiecznych jak filmy czy gazety, albo do celów politycznych, np. symulacje wojny NATO z Układem Warszawskim. Takie programy mogą uwarunkować sposób myślenia młodzieży w kierunku założonym przez twórców.

Podobało mi się natomiast w Polsce powszechne zainteresowanie komputerami wśród młodzieży. Da to na pewno w przyszłości dobre efekty w postaci licznej rzeszy zdolnych informatyków i elektroników.

— Serdecznie dziękujemy za rozmowę i życzymy pomyślnego rozwoju klubu.

— Dziękuję.

rozmawiali:
Jarosław Łojewski
i Piotr Kuzora
członkowie HKK HarcBajt.

I ZNOWU CHOINKA...





Cześć, Maluchy!

Święta Bożego Narodzenia to dla wszystkich polskich dzieci przede wszystkim choinka. A z choinką kojarzą się dwa wspaniałe momenty, ten uroczysty, gdy znajdujemy pod nią prezenty i ten nie tak uroczysty ale równie radosny, gdy ją ubieramy.

Program, który napiszemy dzisiaj w LOGO może być prezentem dla małej siostry lub brata uczących się właśnie liter. Myślę, że okaże się przydatny w trakcie zgłębiania tej trudnej wiedzy. Jeśli nie macie rodzeństwa w odpowiednim wieku, wówczas — wierzę w to głęboko — znajdziecie wśród rodziny, sąsiadów czy znajomych kandydata, na którym wypróbujecie ten program nawet wbrew jego woli.

Pomysł programu jest następujący: na ekranie ukazuje się litera, grający ma znaleźć na klawiaturze taką samą literę i wcisnąć klawisz, na którym jest ona umieszczona. Każda poprawna odpowiedź powoduje zawieszenie na choince jednej bombki.

W pierwszej kolejności musimy postarać się o drzewko. Rozpoczniemy od pojedynczej gałązki. Na drzewku znajdują się zwykle gałązki o różnej długości, musimy więc w ten sposób skonstruować definicję, aby długość uzależniona była od jakiegoś parametru. Jeżeli gęstość igieł porastających gałązki będzie stała, długość będzie jednoznacznie związana z liczbą igieł.

Spójrzmy teraz na naszą definicję.

```
to galazka :igly
 repeat :igly [fd 10 lt 30 fd 8 bk 8 rt
30 fd 10 rt 30 fd 8 bk 8 lt 30]
 repeat :igly [bk 20]
end
```

Żółw porusza się do przodu rysując gałązkę. Co dziesięć kroków odwraca się w prawo lub w lewo o 30 stopni i rysuje igłę o długości 8 kroków a następnie cofa się po własnym śladzie i powraca do poprzedniego kierunku. Parametr **igły** określa ile razy żółw powtórzyć ma sekwencję: prosto, lewa igła, prosto, prawa igła. Wynika więc z tego, że na naszej gałązce igieł będzie dokładnie dwukrotnie więcej niż wynosi parametr **igły**.

Po dojściu do końca gałązki żółw cofa się — również po własnym śladzie — do nasady gałązki. Robi to także etapami po 20 kroków a liczbę tych etapów określa parametr **igły**.

Możemy teraz sprawdzić, jak wygląda nasza gałązka. Wystarczy napisać

cs gałązka

i żółw pracowicie nam ją narysuje. Jeśli coś nie będzie Wam odpowiadać, możecie to bez trudu zmienić, na przykład długość, czy odstęp igieł. Oczywiście będziecie to mogli zmienić także już po napisaniu całej gry.

Skoro umiemy już rysować gałązkę spróbujmy narysować całe drzewo

```
to drzewko
 make "igly 6
 repeat 6 [fd 20 lt 75 galazka :igly rt
75 fd 20 rt 75 galazka :igly lt 75 make
"igly :igly - 1]
 fd 20 rt 1
 repeat 5 [bk 250 fd 250 lt 0.5]
 rt 1.5
end
```

Rysowanie rozpoczniemy od dołu, a więc od najdłuższych gałęzi. Przyjmijmy, dla pierwszych gałęzi parametr igły będzie wynosił 6. Teraz możemy już rysować drzewo. Zauważcie, że zasada rysowania drzewa jest bardzo podobna do rysowania gałązki. Żółw maszeruje do przodu 20 kroków, odwraca w lewo o 75 stopni i rysuje gałązkę, następnie w prawo o 75 stopni, znowu do przodu i lewą gałązka. Różnica polega na tym, że dodatkowo za każdym razem parametr igły zmniejszany jest o wartość jeden.

Druga część definicji to pień drzewa — szerszy u góry a węższy u dołu. Rysujemy go przy pomocy pięciu linii zbiegających się na czubku drzewa.

Teraz możemy już przystąpić do konstruowania samej gry:

```
to gra
 cs ct
 bk 100
 lt 90
 fd 300 bk 600 fd 300
 rt 90
 drzewko
 pu
 rt 155 fd 20
 repeat 6 [pytanie rt 25 bombka lt 25 fd
40]
 bk 250
 rt 50 fd 40
 repeat 6 [pytanie lt 25 bombka rt 25 fd
40]
end
```

Najpierw rysujemy podłogę i drzewko a następnie zaczyna się zasadnicza część gry. Jest ona w programie podzielona na dwie bardzo podobne części. Jest to po prostu zawieszanie bombek na dwóch stronach choinki. Zasada działania programu jest prosta. Komputer przystępuje do procedury pytanie, po jej wykonaniu żółw rysuje bombkę i ustawia się w pozycji do rysowania kolejnej bombki. Zauważmy jednak, że podczas definiowania procedury gry korzystaliśmy z dwóch nieznanych komputerowi procedur, bombka i pytanie.

Z procedurą bombka nie ma najmniejszego kłopotu. Może ona wyglądać na przykład tak:

```
to bombka
 pd
 fd 10 lt 75
 repeat 12 [fd 5 rt 30]
 rt 75 bk 10
 pu
end
```

Nieco trudniejsza jest konstrukcja procedury pytanie.

```
to pytanie
 ct
 make "liczba random 26
 make "liczba :liczba + 97
 make "znak char :liczba
 pr :znak pr "
 make "klawisz rc
 if not :klawisz = :znak [pytanie]
end
```

Najpierw losowana jest liczba od 0 do 25, następnie dodawana do niej jest wartość 97. W ten sposób uzyskujemy jedną z liczb odpowiadających w kodzie ASCII małym literom. Litera odpowiadająca wylosowanej wartości przyporządkowana zostaje zmiennej znak, i wydrukowana na ekranie. Następnie komputer oczekuje, jaki klawisz zostanie naciśnięty. Jeśli zostanie naciśnięty klawisz różny od tego, któremu odpowiada wylosowany znak procedura pytanie wywoływana jest od początku. Zwróćcie uwagę na to, że procedura ta może być powtarzana bardzo wiele razy. Dopiero wciśnięcie odpowiedniego klawisza spowoduje zakończenie wykonywania procedury i przejście do następnej, czyli rysowania bombki (w procedurze gra).

Miłego ubierania choinki, także tej prawdziwej życzy Wam

Romek







KONKURS KOLEJOWY

Swięta to tradycyjny czas podroży: najczęściej podrożujemy koleją. Swięta to także czas wysyłania listow i paczek, a te także podrożują koleją. Nasz przyjaciel, Kubus Literka rozumiejąc trudną sytuację PKP postanowił zostac maszynistą. Od razu stanął przed bardzo trudnym zadaniem. Musi przeprowadzic przez gmatwaninę torow wszystkie wagoniki stojące na bocznicy. Trzeba to zrobic kolejno, po jednym wagoniku, poniewaz dla każdego z nich wyznaczono inną trasę.

Zapewne zauważyliscie, że na kazdym wagoniku umieszczono tabliczkę z dwiema cyframi lub literami. Są to liczby zapisane w systemie szesnastkowym i zawierają zakodowaną informecję o tym, jaką trasą ma pojechac wagonik. Odczytanie tego kodu nie jest trudne. Należy zapisac liczbę w systemie dwojkowym. Kolejne cyfry od pozycji zerowej do siodmej (a więc czytane od prawej strony do lewej) decydują o tym, w ktorą stronę pojechac ma pociąg na kolejnych zwrotnicach-rozjazdach. Jesli będzie to cyfra 0, pociąg powinien pojechac w lewo, jesli 1 — w prawo. Dotyczy to oczywiscie tylko tych rozjazdow, na ktorych tor rozdwaja się, a nie takich, gdy tor zbiega się z innym.

Okreslony sposob przebycia trasy spowoduje koniecznosc zamknięcia kilku zapor na przejazdach kolejowych. Za każdym razem będzie to inna kombinacja. Przyjmując, że zapory opuszczone to 1 a podniesione 0, zapiszmy ustawienie szlabanow podczas jazdy pociągu jako liczbę w systemie dwojkowym (pamiętając, że pozycja zerowa to prawa skrajna cyfra). Numery pozycji odpowiadające poszczegolnym przejazdom umieszczona są na rysunku. Znak odpowiadający otrzymanej liczbie w kodzie ASCII to jedna z liter rozwiązania naszego konkursu.

Założmy, że wieziemy wagonik z liczbą 6B. Najpierw zapiszmy ją w systemie dziesiętnym. Na wszelki wypadek przypomnijmy wczesniej jakim liczbom dziesiętnym odpowiadają cyfry systemu szesnastkowego:

szesnastkowo	dziesiętnie	szesnastkowo	dziesiętnie
0	0	8	8
1	1	9	9
2	2	A	10
3	3	B	11
4	4	C	12
5	5	D	13
6	6	E	14
7	7	F	15

Tak więc liczba 6B przeliczona na system dziesiętny będzie wynosiła

6·16 + 11 = 107

Teraz już bez trudnosci możemy przeliczyc ją na system dwojkowy. Jeden ze sposobow, to wielokrotne dzielenie całkowite liczby przez 2 i zapisywanie na kolejnych pozycjach (od prawej do lewej) 0 jesli dzieli się bez reszty i 1 jesli z resztą.

107/2	53 reszta 1	zapisujemy więc	1
53/2	26 reszta 1	więc	11
26/2	13 bez reszty	więc	011
13/2	6 reszta 1	więc	1011
6/2	3 bez reszty	więc	01011
3/2	1 reszta 1	więc	101011
1/2	0 reszta 1	więc	1101011

Nasza liczba zapisana w systemie dwojkowym ma więc postac następującą.

01101011

Znając tę liczbę możemy już wyruszyc na trasę. O wyborze drogi na pierwszym rozjeździe decyduje cyfra na pozycji zerowej. W naszym przypadku jest to 1, a więc zwrotnica w prawo. Tym samym przejeżdżamy przez przejazd oznaczony cyfrą 0. Zapamiętujemy, że na tym przejeździe zapory winny byc zamknięte, a tym samym na pozycji zerowej naszej nowej liczby wpisujemy — 1. Drugi rozjazd (pozycja pierwsza — 1) znowu w prawo, trzeci rozjazd (pozycja druga — 0) tym razem w lewo. Przejeżdżamy przez przejazd nr 2, a więc i te zapory winny byc zamknięte...

Po przebyciu całej trasy okazuje się, że na przejazdach o numerach 0,2,3,5, i 6 zapory winny byc zamknięte. Zapisując to w postaci liczby w układzie dwojkowym otrzymujemy

01101101

co przeliczone na układ dziesiętny daje:

0·2 + 1·2 + 1·2 + 0·2 + 1·0 + 1·2 + 0·2 + 1·2

0 + 64 + 32 + 0 + 8 + 4 + 0 + 1 = 109

liczbie 109 odpowiada w kodzie ASCII litera m.

* * *

Jesli uda Wam się przeprowadzic bezbłędnie wszystkie wagoniki i odszyfrowac hasło — w co ani ja, ani tym bardziej Kubus Literka nie wątpimy — przyslijcie je na karcie pocztowej z naklejonym kuponem konkursowym (w numerze) na nasz adres: Redakcja „Bajtka", ul. Wspolna 61, 01-687 Warszawa, KONKURS KOLEJOWY. Termin nadsyłania rozwiązań — 1 marzec 1989.

Głowna nagroda w naszym konkursie to — już tradycyjnie — komputer. Tym razem będzie to Commodore 64 ufundowany przez amerykańską firmę Business Computer International.

(b)

32 -
33 - !
34 - "
35 - #
36 - $
37 - %
38 - &
39 - '
40 - (
41 -)
42 - *
43 - +
44 - ,
45 - -
46 - .
47 - /
48 - 0
49 - 1
50 - 2
51 - 3
52 - 4
53 - 5
54 - 6
55 - 7
56 - 8
57 - 9
58 - :
59 - ;
60 - <
61 - =
62 - >
63 - ?
64 - @
65 - A
66 - B
67 - C
68 - D
69 - E
70 - F
71 - G
72 - H
73 - I
74 - J
75 - K
76 - L
77 - M
78 - N
79 - O
80 - P
81 - Q
82 - R
83 - S
84 - T
85 - U
86 - V
87 - W
88 - X
89 - Y
90 - Z
91 - [
92 - \
93 -]
94 - ^
95 - _
96 - `
97 - a
98 - b
99 - c
100 - d
101 - e
102 - f
103 - g
104 - h
105 - i
106 - j
107 - k
108 - l
109 - m
110 - n
111 - o
112 - p
113 - q
114 - r
115 - s
116 - t
117 - u
118 - v
119 - w
120 - x
121 - y
122 - z
123 - {
124 - |
125 - }
126 - ~

KUPON KONKURSOWY

BITY Z LODÓWKI



Ciekawe ilu ludzi zapalając wieczorem światło w swoim domu zastanawia się, jaki procent energii wytworzonej przez spalający się w elektrowniach węgiel lub spadającą na turbiny wodę trafia do żarówki jego lampy? Z pewnością coraz więcej, w miarę jak wzrasta nasz niepokój związany z zagrożeniami ekologicznymi, z deficytem paliw, z coraz bardziej niepokojącymi perspektywami przyszłości rodzimej energetyki.

A jest się nad czym zastanawiać. Każdy kto zetknął się w szkole z prawem Ohma i poznał wzór na ciepło Joule'a — Lenza wie, że podobnie jak to ma miejsce w żarówce tak i w każdym przewodzie płynący prąd powoduje wydzielanie się ciepła — tym mniejszego im mniejszy jest opór materiału, z ktorego wykonano przewód. Nawet jednak w przypadku miedzianych czy stalowych linii przesyłowych przekazywanie energii elektrycznej na duże odległości powoduje „grzanie powietrza" w wielkiej skali. Ponad połowa energii dawanej przez generatory elektrowni znika nim dotrze do naszych domów.

Od początków rozwoju energetyki zajmujący się nią ludzie marzyli o tym, by „oszukać" naturę, zabrać jej ten dotychczas płacony haracz. Marzą także elektronicy — o „kościach" nie wydzielających ciepła, wreszcie specjaliści z dziedziny transportu — o pociągach na magnetycznych poduszkach pędzących z szybkością samolotu. I właśnie u progu tego roku okazało się, że marzenie to wkrótce stać się może rzeczywistością. Kto wie, czy za kilkadziesiąt lat data 1987 nie będzie kojarzyła się z początkiem „ery nadprzewodnictwa".

Zjawisko nadprzewodnictwa, występujące w wielu metalach i stopach, odkryte zostało już przed kilkudziesięciu laty — w 1911 roku. Aż do końca lat pięćdziesiątych fenomen przepływu prądu elektrycznego przez materiał bez oporu nie był praktycznie wykorzystywany. Było to spowodowane warunkami występowania nadprzewodnictwa — mieliśmy z nim do czynienia tylko w ekstremalnie niskich temperaturach, bliskich 0 Kalvinów (minus 273 st.C.). Chociaż w oparciu o osiągnięcia mechaniki kwantowej udało się teoretycznie opisać dość dokładnie mechanizm nadprzewodnictwa nie sposób jest przewidzieć czy może ono wystąpić także przy wyższych temperaturach. Pozostaje metoda „prób i błędów". I właśnie tą metodą u progu 1987 roku dokonano przełomu, o którym mówi się dziś nie tylko w światku fizyków.

Alex Muller i Georg Bednarz — pracownicy laboratorium IBM (tak tego samego!) w Ruschlikon koło Zurichu badali izolatory ceramiki powstałe na bazie tlenków baru i miedzi. Zajmujący się tymi samymi materiałami chemicy francuscy odkryli wcześniej, że mają one wiele cech wspólnych z metalami. Muller i Bednarz idąc tym tropem postanowili zbadać, czy nie daloby się wywołać w nich efektu nadprzewodnictwa. Dało się i to w temperaturze minus 243 stopnie — o 20 większej niż w dotychczas badanych materiałach!

Rozpoczął się prawdziwy wyścig. Niemal co kilka dni prasa popularna Stanów Zjednoczonych, Związku Radzieckiego i innych krajów meldowały o pokonaniu kolejnej granicy występowania nadprzewodnictwa — minus 200, 190 stopni, 180 stopni. Te ostatnie rezultaty miały już wymiar nie tylko naukowy, ponieważ temperaturę minus 190 stopni otrzymać można stosunkowo łatwo przy pomocy ciekłego azotu-gazu powszechnego i znacznie tańszego od helu, dotychczas używanego do schładzania nadprzewodzących próbek. Kiedy 18 marca 1987 r. w nowojorskim „Hiltonie" zwołano konferencję naukową na temat nadprzewodnictwa salę obrad szturmował już od rana kilkutysięczny tłum fizyków. Gdy podczas obrad tej konferencji Bertram Batlogg z Laboratorium Bella wyjął z kieszeni kawałek giętkiej taśmy obwieszczającej „nasze życie się zmieniło" poważni naukowcy wiwatowali jak nastolatki na koncercie „Europe". Ta taśma to był właśnie materiał nadprzewodzący w temperaturze ciekłego azotu.

Mała grudka ceramicznego spieku lewitująca w powietrzu nad magnesem — taki pokaz zobaczyć było można i w warszawskim Instytucie Fizyki PAN. Prezentujący ją dr Piotr Przysłupski włączył się do światowego wyścigu fizyków i nawet przez kilka dni przewodził stawce tych, ktorzy opracowali „najcieplejszy" nadprzewodnik. Gra idzie zresztą nie tylko o wynik. Trzeba mieć materiał o jak najwyższej temperaturze krytycznej by w w ciekłym azocie utrzymać nadprzewodzące właściwości materiału nawet wowczas, gdy przepuści się przez niego stosunkowo duży prąd.

Rok 1987 uczeni pracujący nad nadprzewodnikami wysokotemperaturowymi zamknęli nie lada sukcesem — Nagrodą Nobla dla Mullera i Bednorza. Kolejny rok nie przyniósł już tak spektakularnych sukcesów. Rekord najwyższej temperatury dzierży dziś laboratorium San Jose w Kalifornii — minus 140 st.C. Jednak Paul Grant, pracownik tego laboratorium optymistycznie przewiduje, że już wkrótce dzięki wyrafinowanej technologii tworzenia cienkich warstw nadprzewodzących osiągnięty zostanie znacznie lepszy rezultat — około minus 73 st.C. A stąd już tylko krok do mikroelektroniki nadprzewodników. Już dziś Japończycy opanowali ponoć produkcję pamięci nadprzewodzących o pojemności 1 Kb. Może to i niewiele w zestawieniu z możliwościami „kosci" krzemowych, jednak tradycyjne pamięci nawet te wykonane z arsenku galu, nie mogą się nawet równać z nadprzewodzącymi pod względem szybkości działania.

Perspektywa stworzenia zupełnie nowej generacji komputerów z jednej i znacznego obniżenia zapotrzebowania świata na energię elektroniczną oraz zastosowania z pomocą nadprzewodników syntezy termojądrowej, z drugiej strony — to już choćby zapowiedź, jak „epoka nadprzewodnictwa" u progu której stoimy różna będzie od dnia dzisiejszego.

Grzegorz Onichimowski